Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

FÁBIO BLESSA FERNANDES LINO

DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE BOTNETS

DEVELOPMENT OF A BOTNET DETECTION SYSTEM

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

FÁBIO BLESSA FERNANDES LINO

DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE BOTNETS

DEVELOPMENT OF A BOTNET DETECTION SYSTEM

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Prof. Doutor Paulo Salvador e do Prof. Doutor António Nogueira, Professores Auxiliares do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais pela força e motivação que me levou a completar esta dissertação.

o júri / the jury

Presidente / president

Doutor Aníbal Manuel de Oliveira Duarte
Professor Catedrático da Universidade de Aveiro

Vogais / examiners committee

Doutor Rui Jorge Morais Tomaz Valadas
Professor Catedrático do Instituto Superior Técnico

Doutor Paulo Jorge Salvador Serra Ferreira
Professor Auxiliar da Universidade de Aveiro (orientador)

Doutor António Manuel Duarte Nogueira
Professor Auxiliar da Universidade de Aveiro (co-orientador)

agradecimentos

Gostaria de agradecer aos professores Paulo Salvador e António Nogueira pela paciência que demonstraram ao longo do desenvolvimento desta dissertação, por estarem sempre dispostos e prontos a ajudar. Como não tive muito tempo livre, não conseguir acabar esta dissertação no tempo planeado.
Agradeço também ao Nelson, pelas dicas relativas a sysadmin e PHP que não se reflectiram somente nesta tese mas para um enriquecimento geral do meu conhecimento. Obrigado a todos os meus colegas que me ajudaram a desenvolver e corrigir este documento.
Obrigado a minha Maria pela paciência ou impaciência ☺ ao longo de todos os momentos, todos os seus conselhos e dicas foram importantes.
Muitíssimo obrigado aos meus avós, que torcem e rezam por mim, por tudo o que me ensinaram.
Claro que não poderia faltar os meus pais, que sempre me guiaram ao longe de todos os momentos da minha vida e que sem eles não seria metade do que sou. Obrigado pelas lições de vida, principalmente por me ensinarem que tudo é possível desde que se queira e que para tudo existe uma solução. Obrigado mãe e pai por toda a alegria, motivação, amor e compreensão! Obrigado!

palavras-chave

Botnet, redes neuronais artificiais, sistema de detecção de intrusões, tráfego ilícito, malware.

resumo

O tráfego ilícito é um dos maiores problemas da segurança em redes. É necessária uma estratégia global contra esta ameaça, uma vez que este problema pode afectar a economia gerada pela internet a um nível global nos próximos anos. As técnicas tradicionais de detecção de computadores *zombie*, como *antivírus*, *firewalls* e *anti-spywares* não são eficientes contra esta ameaça. A principal razão para este fracasso é a limitação imposta pelas metodologias tradicionais face ás novas ameaças que constantemente aparecem. Para ultrapassar esta limitação, propomos uma nova abordagem diferente dos actuais sistemas de detecção de intrusões, que conjugada com os métodos tradicionais pode garantir um nível elevado de segurança. Esta nova abordagem é baseada na identificação de padrões de tráfego de rede. Cada serviço de rede tem uma característica que o define e, sendo assim, podemos tirar partido desse facto para identificar o tráfego ilícito correspondente a *botnets* e outros *malwares*. Para identificar o que é tráfego ilícito e o que é lícito, é usada uma Rede Neuronal Artificial que é treinada para identificar os padrões de tráfego de rede correspondentes ao tráfico ilícito. Depois de identificado o tráfego ilícito, o sistema proposto neste trabalho vai gerar alarmes que alertarão o administrador do sistema em caso de identificação de computadores *zombie* ou infectados. O próximo passo será tomar uma medida preventiva, que pode ir desde bloquear o endereço IP correspondente a essa máquina até colocá-la sob um nível de vigilância extra. Os resultados obtidos mostram que esta metodologia de identificação de tráfego ilícito é uma técnica possível de ser usada no dia-a-dia devido à sua elevada taxa de identificação e à sua baixa carga computacional. Esta técnica pode identificar problemas actualmente indetectáveis pelas metodologias vulgarmente usadas.

keywords Botnet, malware, intrusion detection system, illicit traffic, artificial neural networks.

Abstract

Illicit traffic is one of the major issues in network security. A strategy for a global partnership against it is needed in order to avoid illicit traffic from becoming a serious threat to the Internet economy and to global security in the forthcoming years. Traditional Zombie detection techniques, such as antivirus, firewalls and anti-spyware are not effective against this threat. The main reason for this failure is the limitation of these traditional methodologies to detect new threats. To overcome this limitation, we propose a new intrusion detection approach that works together with traditional methods in order to ensure a higher level of protection/security. This new approach is based on the identification of traffic patterns. Each network service, as well as illicit traffic corresponding to botnets and other malwares, has a characteristic traffic pattern that can univocally identify it. In order to decide which network traffic is illicit or licit, we use an Artificial Neural Network that is trained to identify the illicit patterns. After identifying the illicit traffic, the proposed system will generate alarms to the system administrator in order to alert him about a Zombie or an infected computer. After this identification phase, the system administrator can take a security action, like blocking the corresponding IP Address or putting it under a deeper surveillance. The results obtained show that this is a feasible and efficient methodology, since it provides high detection rates with low computational overhead. So, we believe that this methodology can be an emergent technique that will deal with untraceable threats that current methodologies are unable to deal with.

Contents

Index of figures

Index of tables

Acronyms

RHEL	Red Hat Enterprise Linux
ANN	Artificial Neural Network
GUI	Graphical User Interface
DWH	Data Ware Housing
SCP	Secure Copy
BoNeSSy	BotNet System Security
HTTP	HyperText Transfer Protocol
FTP	File Transfer Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
RPM	Red Hat Package Manager
FANN	Fast Artificial Neural Network
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
DDoS	Distributed Denial of Service
BPN	Back – Propagation Networks
MLP	Multilayer Perceptions
API	Application Programming Interface
URL	Uniform Resource Locator
SQL	Structured Query Language

1 INTRODUCTION

Illicit traffic is one of the major issues in network security. A strategy for a global partnership against it is needed in order to avoid illicit traffic from becoming a serious threat to the Internet economy and to global security in the forthcoming years [1]. Illicit traffic can be defined as all data flows that are generated with the intention to disable, destroy, control or steal systems information.

The Internet exponential growth, the massive increase in the number of provided services and the heterogeneity of network technologies makes it very difficult to intercept and identify illicit traffic. This threat can affect organizations, countries, governments and normal users that, without their knowledge, can participate in DDoS attacks, have their private information published, exposed or sold to companies that use this kind of data to focus their advertisement strategies, or their whole system compromised. The costs of temporary or permanent damages caused by intruders' unauthorized access to networks and computer systems have urged different organizations to implement various systems that monitor data flows in their networks. Intrusion Detection Systems (IDS) aim to protect networks and computers from malicious network-based or host-based attacks. This type of software can be invisible and untraceable to any actual IDS with default detection methodologies.

Nowadays, a high percentage of computers are infected with some kind of malware. This means that current illicit traffic detection methods are not efficient, being only limited to block some of this data. Although there are some responses to this global threat that were proposed by specific communities, they are basically small fragments of a global solution. So, malware appears to be always one step ahead of detection systems.

As security systems get more complex, malwares in general get more intelligent. When a new system is built and a strong security system is incorporated (Intrusion Detection System), it's just a matter of time until some kind of malicious program overpasses its security and completely exposes all the system information.

Conventional methods to detect and map illicit traffic are getting more and more inefficient as time passes. It is not too difficult to mask network traffic. There are simple, easy and effective ways to do it. So, traditional techniques to detect illicit traffic are easily bypassed with a simple traffic camouflage to confuse and hide its suspicious actions. Some security systems check the port where applications are running in order to identify and eventually block illicit traffic (*misuse* approach). In this situation, any malware can simply change its functioning port, erasing all its traces and becoming invisible to this kind

of security systems. Another method used by security systems is to inspect all the network traffic that passes through them and verify their signatures, which are already known. However, this can be a hard task since traffic can be encrypted. Data flow analysis based on signatures can affect network stability, since it is necessary to inspect and process all the traffic that passes through in order to match the corresponding signatures.

Semantic and syntactic analysis of network traffic avoid some of the problems presented by the above methodologies, but this kind of analysis also requires a lot of computational load and is not appropriate when confidentiality is required.

It is not possible to build a bulletproof security system, but we can get very close. What we propose in this dissertation is a new methodology to identify and stop the propagation of illicit traffic when conventional tools are not able to do that by their own. So, if we use both approaches we can get a very robust security system.

The proposed system is based on the collection of flow statistics, such as packet sizes and number of packets that pass through the network, and their corresponding association to each IP address. While traffic is being captured, it is also being sent as input to a pre-trained neural network that will analyze all the patterns and try to discover new threats. In order to be more effective discovering these new threats, the artificial neural network needs to be trained again, that is, it needs to be periodically trained. This data is captured using probes that are strategically located on the network. As the neural network processes all input data, its output will be positive or negative according to the data patterns. This output will be stored and the user will be able to monitor his neural network through a fancy graphical user interface that will provide extra options to personalize the system in order to meet the user and network needs. The possibility to define and trigger countermeasures will also be provided for a fast system healing.

Neural networks are a promising technique that has been used in many classification problems. Our Network Intrusion Detection System (NIDS) will also be based on this approach.

All traditional methods for illicit traffic detection are well explored and widely documented, so they can be very easily overpassed. As malwares evolve, intrusion detection systems must at least follow the same level of evolution, otherwise IDS won't be effective enough to block and discover new threats. The same applies to malware techniques and infection methods: a good IDS should be able to block them independently of the infection method.

Our approach can block new threats independently of their camouflage method, since it does not look for well-known identification techniques but for traffic patterns that

are generated by these malicious applications. This new approach should be used together with standard identification methods in order to maximize the system security.

Nowadays, competition between companies to develop new projects that can beat the crises and take over the market has increased. Data security is extremely important, since projects, ideas and source codes are all stored in version control systems and if any bad intentioned person has access to these repositories all the company's private information and data can be easily stolen and all the projects can be compromised. This kind of information can worth millions of euros to companies and their security can be decisive to know whether they will accomplish their financial and market goals or not.

From my point of view, this kind of product can be easily sold to big, medium or small companies that want a more reliable security control. In addiction to the whole product, several plug-ins or add-ons with extra reporting or detection improvements can be developed according to the customer specification. An example of an add-on can be the addiction of extra functionalities to the neural network so it will not only identify malware but other kinds of traffic that the customer wants to ban from his network.

1.1 Objectives

This dissertation intends to develop a network intrusion detection system based on an artificial neural network. This system will be sufficiently open so that new add-ons or plug-ins can be integrated to easily expand the program capabilities.

There will be several development phases:

- Research of current tools and detection methods;
- Development of tools to capture, store, aggregate and treat all data that is suitable to be used as input of the neural network;
- Develop a neural network model, based in an existent library, fully dedicated to find anomalous network traffic patterns;
- Analyze and design a database scheme that will be used to store all data;
- Develop a graphical user interface (GUI) to easily monitor the system status;
- Integrate all the system modules, so that the system can work as a whole.

1.3 Document Outline

This report is organized in six chapters and each chapter is structured into sections and eventually sub-sections. The document structure is summarized here:

- Chapter 2 presents the background of this dissertation, giving an overview of some basic knowledge that is needed to fully understand all concepts present in this work. Also, provides a brief overview of existing works and tools used for data monitoring, capture and analysis.

- Chapter 3 discusses the big picture of this work as well as all its main development phases, from discussing the system architecture, data capture method, database model, system integration and management user interface.

- Chapter 4 presents the system test and validation. Here, all results and graphics from the neural network tests will be presented and discussed.

- Chapter 5 presents the conclusions of the developed work.

- Chapter 6 points out some topics for future work.

2 BACKGROUND

This chapter presents the background for this dissertation, briefly mentioning some important aspects that will be developed during this work and clarifying some ambiguous definitions. It also focuses on some adopted techniques for identifying illicit traffic.

This chapter also presents a small overview of the environment that involves each development phase, giving also a big picture of all the theoretical concepts that are needed for the dissertation.

2.1 Artificial Neural Networks

Human brain has many advantages when compared to computers. For computers to solve problems, they need to be pre-programmed to know *a priori* all possible inputs in order to know exactly what to do once they face each one of them.

Artificial intelligence is a solution for this kind of problems, bringing the computer ability to solve problems to another level and making learning possible. Artificial Neural Networks (ANNs) can be compared to biological neural networks, they can be considered as an emulation of the biological model. The computational area that studies this approach is known as neurocomputing [2]. The usage of neural networks brings some advantages and disadvantages: some of the most important advantages are being able to perform tasks that a linear program cannot, learning without being reprogrammed and having a lot of applications in areas as diverse as finance, economics, medicine, engineering, geology, physics and biology [3]; the main disadvantages are the fact that they need to be trained before they can properly work and generally require high processing times (at least, for large neural networks).

ANNs are associated to learning methodologies. These machine learning techniques can be usually divided in two main groups: the supervised and the unsupervised models. By supervised learning we mean all training procedures that receive a feedback on how good their performance is during training. In unsupervised learning, there is no performance feedback available [4].

As the biological model, the ANN is mainly composed by interconnected artificial neurons organized in different layers. Each connection has a specific associated number (+ or -) that determines the influence of the previous neuron output in the current neuron.

An artificial neuron can be implemented in different ways. The general mathematical definition of a neuron (excluding input neurons) is shown in equation 1 [5]:

$$y(x) = f\left(\sum_{j=0}^{n} w_{i,j} x_j\right) \tag{1}$$

where

i => Present node

j => Node from where the output came

x => Represents the neuron itself

$y(x)$ => Output axon

$w_{i,j}$ => The connection weight

$\sum_{j=0}^{n} w_{i,j} x_j$ => Linear weighted sum that transforms multiple inputs in a single output.

f => Activation function, usually non-linear, can be interpreted as a conditioning function

The following image (figure 1) shows a diagram of the above formula for a better understanding:

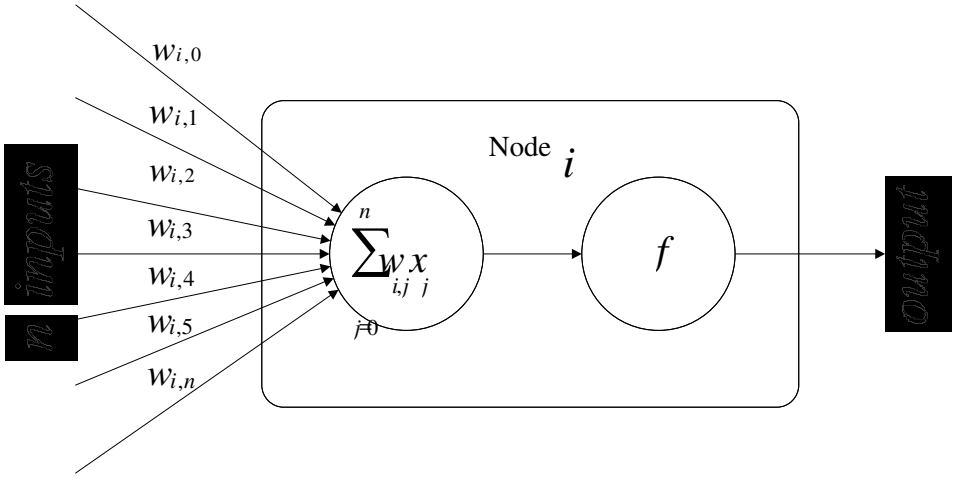

Figure 1 Diagram of a single artificial neuron

An ANN can have a specific number of inputs that depends on the input data. Input data should be normalized in order to get better results [6] and then handled by the input neurons, that are considered to be the first layer of neurons. The output values are given by the output neurons: so, if a problem is expecting a single value, it must have a single output neuron. In other words, the number of input and output neurons is determined by the data characteristics. The hidden layers connect the input and the output layers: in our case, we will use only one hidden layer, which can have as much hidden neurons as the user wants. The number of hidden neurons has to be carefully specified, as the level of abstraction, the speed and the learning capability depend on this value. Over-fitting is a common error when using ANNs where the number of hidden neurons is overestimated, which degrades the generalization capability. A big picture of and ANN is shown at figure 2 [7]:

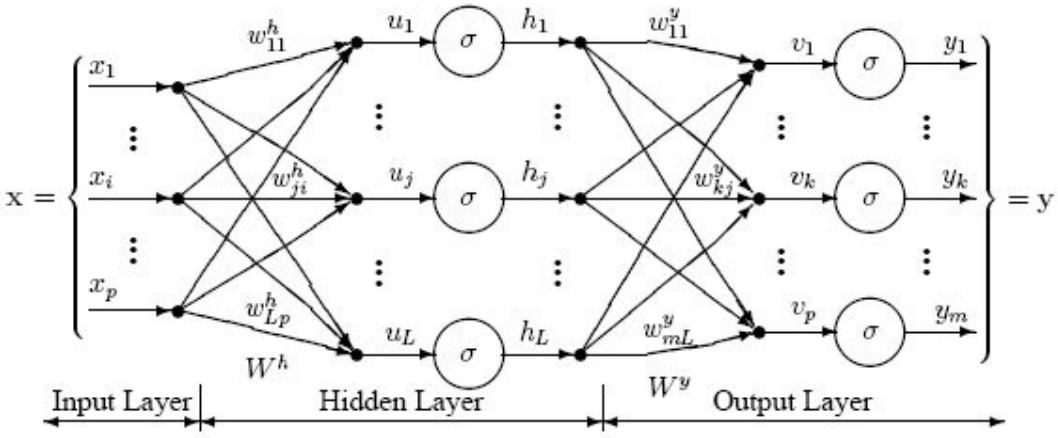

Figure 2 Diagram of an entire ANN

There are two ANN fundamental models: Multilayer Perceptrons (MLP) and Back-propagation Networks (BPN). A brief description will be given for the MLP model during this work, since only BPN training algorithms will be used. MLP is one of the earliest and perhaps simplest models that are capable of representing any Boolean function and of approximating more general functions to an arbitrary precision level [4].

Rigorously speaking, BPN is a learning algorithm and not a type of neural network, but as BPN is primarily used with one type of network it is convenient to refer to this kind of network as Back-propagation Networks. These kinds of networks are characterized by

being feedforward, having multilayer perceptrons and supervised learning. BPN is known as being very good in the recognition of complex patterns [8].

There are a wide variety of learning algorithms that we can use to train our ANN. The algorithms that we introduce now are the ones that will be used to train and test the performance of the ANN that will be used to match the illicit traffic patterns.

INCREMENTAL

In the standard backpropagation algorithm the weights are updated after each training pattern. This means that the weights are updated many times during a single epoch. For this reason, the neural network will train very rapidly for some problems but will take a lot of time to train for more advanced problems [5].

BATCH

This is also the standard backpropagation algorithm where the weights are updated after calculating the mean square error for the whole training set. This means that the weights are only updated once during an epoch. For this reason, some problems will train slower with this algorithm. But since the mean square error is calculated in a more correct way than in the incremental training case, some problems will reach a better solution with this algorithm [5].

RPROP

This is a more advanced batch training algorithm, which achieves good results for many problems. The RPROP training algorithm is adaptive and does not use the learning_rate parameter [5].

The RPROP training algorithm is described in [9], but the actual learning algorithm used here is the iRPROP training algorithm, which is a variant of the standard RPROP training algorithm that is proven to be slightly better [10].

QUICKPROP

A more advanced batch training algorithm which achieves good results for many problems. The quickprop training algorithm uses the learning_rate parameter, in contrast with the previous algorithm, along with other more advanced parameters, but the change of these advanced parameters is only recommended for users that have a deep insight in how the quickprop training algorithm works ,since its efficiency can drop for badly chosen parameters [5].

Artificial Neural Networks are highly configurable: depending on the input data or the expected output, we can decide which are the best parameters' configuration that will give an optimal result.

2.2 Malware

Malware or malicious software, as already stated above in this dissertation, is any kind of software that gets itself installed in a machine without its owner permission and performs a variety of actions from just annoying the user (like pop-up propaganda) to causing serious and severe damage to its host (stealing identities, passwords, disabling security, erasing data, among others).

It is difficult to categorize all these illegal software since its definition varies from author to author. Some authors divide it in 3 groups: malware, crimeware and grayware [11]. Other authors use only two main groups: the grayware and the malware, that includes the crimeware [12]. Taking the last approach into consideration, the first group can be classified as all applications that behave in a manner that is annoying or undesirable, reducing the computer performance, poping-up some extra advertisements or simply disturbing the user. However, these problems are less serious or troublesome than malware, which is oriented to harm the computer and steal personal or confidential data, between other malicious actions. Unfortunately, most of the malware was developed to perform illegal activities.

Recent malware is not anymore developed to be intrusive or to damage the computers; instead, it is designed to be low profile, so all his activities can continue under the radar.

There are several categories or types of malware, from the most known form – virus - to wabbits, the rarest kind of malware. As there are several types and names of malware, it is complicated to define a barrier between one definition and another. The following text, describes some of the malware categories and types, giving a brief overall overview:

- Virus is a small computer program that alters the machine behavior without permission or knowledge of the user and relies on someone or something else to propagate from one system to another. A virus has to be self executable and must replicate itself. Normally, it is programmed to do some kind of damage in the computer, but some are just made to spread. Viruses are reducing their quote in malware's total infection percentage, since they are made only to harm the

computer and do not take any advantage of its data. There are several types of viruses, ranging from file infector viruses to master boot record viruses, for example. [13]

- Worms are programs that replicate themselves from machine to machine without the use of a host file, or in other words, they propagate by themselves without the need for outside assistance. This is the big difference between worms and viruses. This kind of software can be considered as a sub-class of virus. [13]
- Wabbits or rabbits are more annoying than malicious; they replicate themselves in the same computer, but not through the network or mobile device such as viruses or worms. Nevertheless, they can have malicious purposes, like consuming all system memory by constantly forking themselves. [13]
- Trojans, as the name suggests, are programs that look to be legitimate or useful software. The user usually trusts on its source, but they are tricked with a program that installs itself in the machine and can perform malicious actions (that's why it is named with the mythological *Trojan Horse* designation, that was offered by the Greeks to take over Troy). This malware can perform a series of illicit actions, like for example: steal all passwords present in the system, log all keyboard activities, download/upload files to the infected PC, perform screen captures, disable system functions, enable webcam share, among other specific functions. The two most famous Trojans are *Netbus* and *Subseven*. [13]
- Adware is a form of advertising that is also intrusive and in some instances criminal. (The term Adware is used to denote a certain type of intrusive advertising but is also the registered and trademarked name of a company that sells anti-spyware and anti-spam software.) Adware takes the form of popup advertisements or banner ads and comes from several sources. One source is certain "freeware" or "shareware" products that vendors give away without charge but have embedded advertising into the code so that they suddenly appear upon the screen and may or may not be removable, or could be subject to a time-delayed closure. As a condition of use, some adware programs go on to install programs on the user's machine, known as Adbots, and these programs then act as a type of Spyware sending back information to their advertising source pertaining to the user's behavior. Often these additional installations are undisclosed. Illicitly installed software may redirect web browsers to access all pages through a proxy, which can add banner advertisements to totally non-commercial pages that in reality bear no such content. Home-page hijacking is also a common problem: a

web browser's default or home page may be reset to a commercial site, so that, every time the web browser is started up, it shows advertisements or installs additional adware. [14]

- Spyware, as almost all malware, is a program installed in computers without the user knowledge or permission. It can change some configurations on the machine, collect user information, like tracking visited websites, reporting all this "stolen" information through an internet connection or an autodialing modem to a third party. [14]
- Backdoors are methods to bypass system security in order to gain access without proper authentication through bugs in programs or operating systems. So, normally a program that uses this technique to infect the user's computer, namely to install malware, is called backdoor. [13]
- Exploits are pieces of code or sequences of commands that take advantage of a system or program vulnerability, glitch or bug. Not necessarily malicious, it can be used by security researchers to show how vulnerable a system is. When used by bad intentioned people, exploits can be very dangerous and their effects can be devastating as they can explorer many security holes, like for example bypass authentication, corrupt systems (buffer overload), download and upload protected files, among others.
- RootKit takes the concept of a Trojan horse to a much more powerful level. Although the name implies something else, RootKits do not allow an attacker to gain "root" (super-user) access to a system. Instead, RootKits allow an attacker who already has super-user access to keep that access by foiling all attempts of an administrator to detect the invasion. For example, to hide a file the RootKit intercepts API calls from explorer to keep certain files hidden from display or giving a wrong number of files in a folder. RootKits consist of an entire suite of Trojan horse programs that replace or patch critical system programs. The various tools used by administrators to detect attackers on their machines are routinely undermined with RootKits. Rootkits can also be used for legitimate purposes by vendors to control software licenses and copyrights or by antivirus for self defense against any malicious software. [15]
- Keyloggers are programs capable of logging and noting all keyboard activities without the knowledge or permission of the user. It can act alone or as part of a Trojan or any other malware. The logged activities can be sent to the criminal

through FTP, SMTP or any IM service protocols. There are several different keylogging techniques, from hardware to software [11].

- Dialers are programs illegally installed in a computer that has the ability to make phone calls from the computer modem (if it is installed and connected to the phone line). This malware can call to paid site numbers, without the victim permission, so that the victim will be charged for these calls [11]. This is an old virus type, since nowadays, most of the internet users use broadband internet accesses.
- URL injectors are programs that modify the browser's behavior regarding some or all domains. They modify the URL that is submitted to the server to profit from a specific domain. This is often transparent to the user.
- Phishing has become a very widespread problem for Internet users. A criminal organization or individual sends out emails that are an exact copy of legitimate emails from a well-known company that many random recipients are likely to have accounts with. The emails look exactly like official emails from that company, but direct the recipient to a web site for some vital purpose. Often the email will warn that an account is about to be involuntarily closed, or a large bill is about to be referred to a collection agency, and this is the customer's last opportunity to do something about it. The web site that the customer is referred to will have an address (URL) that seems to be right for the company, and the content will be carefully set up to duplicate the company's real web sites. Of course, it will be completely controlled by the criminal ‘‘phishers." Victims may be made to feel more comfortable by the fact that they have to log in using their pre-established username and password, but, of course, the site is not checking their identity, it is simply recording the user-names and passwords that are entered. That might be all that happens; stealing a customer's user-name and password is often all that is required for great financial gains, in which case the customer will quite probably be redirected to the company's real web site with a message saying that the password entered was incorrect. Otherwise, the illicit site may continue to ask for an ever-widening variety of personal details, account numbers, and anything else the customer can be persuaded to divulge. The word ‘‘phishing" is just a respelling of ‘‘fishing," which is really what it is, fishing for information. The change of ‘‘f" to ‘‘ph" is just a strange modern fad of no significance. [14]
- Bots and Botnets will be extensively discussed in the following chapter.

- Browser Hijacker or hijackware is any program designed to change a computer browser settings, search results, home page, error message pages, bookmarks or other unwanted or unexpected content [11].

The first objective of malware is to infect their victims. The infection method can vary according to its main purpose, its type or the developer experience. After this phase, this malicious program has to protect itself from security mechanisms that can discover it and block its activity. To protect itself, it adopts several advanced or basic techniques, like [16]:

- Packers and cryptors: the use of encryptation and compression software;
- Server-side polymorphism: generate new version through re-packaging or re-encryptation so only the appearance changes, not the inside;
- Code obfuscation: this method is getting more popular among the new malwares and it consists in the insertion of junk or meaningless instructions between the real ones;
- Anti-debugging and anti-emulation: this method is used to trick debuggers and emulation programs. If the malware detects that debugging programs are being run, the malware can quit itself or change its execution course;
- System monitoring: execution environment is taken into account here. The malware monitors any system changes that can affect its right functioning.
- Stealth: this technique is used by almost all malwares and can go from hiding its files, hooking relevant APIs, among other behaviors.
- Anti-analysis: an effective way to discover if the malware is being run in test or lab environment or honeypot systems.

2.3 BotNets

The wide use of computers and the network growth has open new doors, new opportunities for hackers to expand their infections by multiplying the number of attacks. With this huge number of computers worldwide, can you imagine all the processing power and bandwidth that a small percentage of these machines can provide if they work for the same purpose?

Botnets can be simply defined as a collection of compromised computers that are used by some individual or organization, named as botmaster, without the awareness or the authorization of their owners, usually for nefarious purposes [17].

Computers that are taken over by secret, hidden software planted by hackers and scammers often become part of a robot network.

This organized bots can be naively compared to computer clusters, but of course, focusing its computational power to illicit activities. Bots are generally and more typically created by finding vulnerabilities in computer systems, exploiting these vulnerabilities with malware, and inserting malware into those systems. The bots are then programmed and instructed by the botherder (another synonym for botmaster) to perform a variety of cyber attacks, including attacks that involve the further distribution and installation of malware on other information systems. Malware, when used in conjunction with botnets, allows attackers to create a self-sustaining renewable supply of Internet-connected computing resources to facilitate their crimes (see figure 3). Some of the malware discussed earlier in this report is distributed using botnets. There is thus a cyclical relationship: malware can be used (are indeed used) to create botnets, and botnets can be used (and are indeed used) to further distribute spam and malware. Figure 3 demonstrates the relationship between malware and the botnet lifecycle. When malware infects an information system, two things can happen: something can be stolen (e.g. information, money, authentication credentials etc.) and the infected information system can become part of a botnet (eg. to perform distributed attacks). When an infected information system becomes part of a botnet it is then used to scan for vulnerabilities in other information systems connected to the Internet, thus creating a cycle that rapidly infects vulnerable information systems [1].

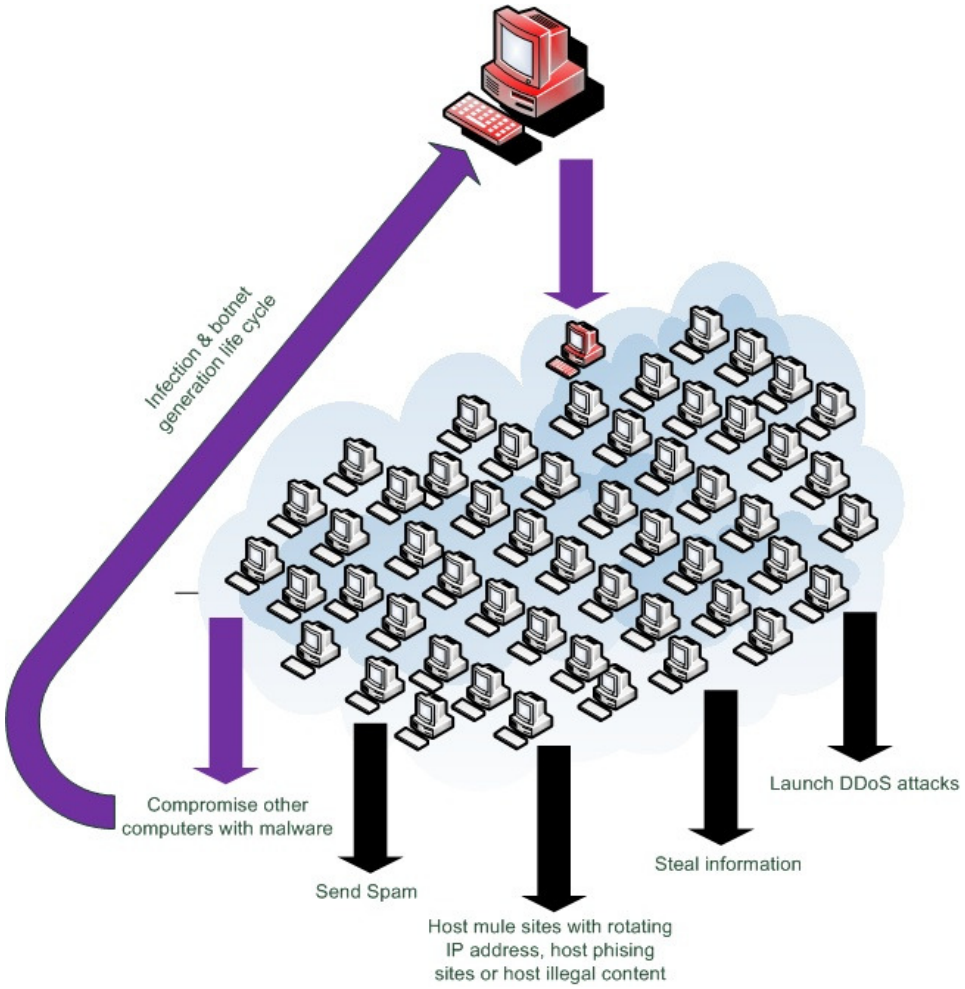

Figure 3 Botnet life cycle (modified from [1])

A perfect example of this life-cycle is the existence of some botnets that are used to disseminate spam in social network sites. As recently happened [18], a malware named Koobface was responsible for thousands of infections in social network users, like users from Facebook and Myspace, creating an enormous botnet. The trick of this malware is that it is able to break the captchas code, even the most advanced ones, not using character recognition programs but using cheap labor offering a reward of more or less 1 cent of Euro to each image code that is correctly typed. With this, the only defense that the sites had to fight the bots was broken and thus a mass infection begun.

The infection method used by this malware was simple and efficient: every bot PC sent a message to other users, looking like a youtube video, and after the victim simply clicked the play button to see the video, a message was prompted warning about the necessity to install a video codec. After the download of the assumed video-codec, that

PC was made part of the botnet sending more messages to other users in the social network site.

There are several reasons for a botmaster to focus the use of his botnet: to make his botnet grow - malware multiplication (as above explained), to send spam, to perform DDoS attacks, to generate traffic (commercial reasons), for rental purposes, for sniffing, keylogging, identity theft, phishing, among other illicit activities.

Botnets are not anymore owned by people that want to make some “noise” or to be famous. Botnets are more and more associated with cyber-gangs that join their efforts to have a strong impact in society. A malware can install a bot plus other illicit software from a third party creating here some kind of relationship between cyber-gangs.

2.4 Intrusion detection and countermeasures systems

In this sub-section we are going to give a brief introduction about Intrusion detecting systems (IDSs) and its countermeasures.

Computer systems are getting more powerful and because of that, more complex to configure, harder to explore and monitor and more suitable for bugs (exploits). Knowing this, IDSs play an important role in monitoring and enforcement the enterprise security.

A short definition of IDS could be: “Intrusion detection is the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems” [19]. An IDS can be categorized by a range of different characteristics, from its monitoring strategy, its detection technique and its architecture.

Regarding the detection technique, we can categorize IDs in two main groups [20]:

- Anomaly detection attempts to model normal behaviors so any events that violate this model are considered to be suspicious.
- Misuse detection tries to model abnormal system behavior or system abuse to identify attacks. This technique filters event streams and tries to search for illicit activities by identifying their patterns. That is the reason why this approach is known as a pattern-matching technique.

If we look at its monitoring strategy, we can divide IDSs into four groups [19]:

- Host-based monitors collect data internal to the PC, usually at the operating system level. The sensor normally consists of a software agent that monitors all host activities by analyzing system calls, logs, file system modifications and others.

- Network-based monitors have devices running in promiscuous mode located at strategic points inside the network so that they can capture network packages to be analyzed *a posteriori*. They are known as NIDS.
- Application-based monitors collect data from running applications. The data sources include applications' event logs and other data that is internal to the application.
- Target-based monitors use cryptographic hash functions to detect changes to system objects and then compare these changes to a policy.
- An hybrid intrusion detection system combines two or more approaches. For example, host agent data is combined with network information to form a comprehensive view of the network.

Regarding their architecture, we can also split IDSs into four categories [20]:

- Monolithic systems are considered to be a simple all-in-one application that focuses on a specific host or system. Such systems are conceptually simple and relatively easy to implement.
- Hierarchic systems consist in a bunch of subsystems reporting hosts activities to a global system. The use of a centralized controller unit allows information from different subsystems to be correlated, potentially identifying transitive or distributed attacks.
- Agent-based systems are more recent models where the architectural design divides the system into distinct functional units. These may be distributed across multiple systems with its probes reporting to monitors, which then report to a higher level.
- Distributed system is an approach that promises high scalability and has the potential to recognize widely distributed attacks.

There are several issues that an IDS should handle, regardless of its architecture or detection technique, in order to be a good detection system:

- Effectiveness is the most important requirement: an IDS must be able to accurately and consistently detect attacks or other predefined usage patterns of interest [19]. In other words, it must be difficult to fool.
- Easy to use and configure to the system it is monitoring: a well-designed tool that can enable a non-security expert to reliably perform routine security-related tasks can help customers deal with scarcity of security experts [19].
- It should adapt to its environment and staying current with the system as it changes, as new applications are added, as the system is upgraded and updated,

or if any other modification occurs. The system profile will change over time and the IDS must be able to adapt itself [15].

- The system should run reliably with provision for redundant or other fault-tolerant features [19]. It must survive a system crash and not have its knowledge-base rebuilt at restart [15].
- Performance and Efficiency are critical: the system should be able to make optimal use of computing, storage, and communications bandwidth and to perform the monitoring tasks for the target environment without falling behind [19]. It must impose minimal overhead on the system [15].
- The system must be safe: it must have features that prevent unauthorized people from using it as a vehicle for attacking others [19].
- It must be difficult to sabotage: the system should be self-healing in the sense that it should monitor itself for suspicious activities that might signify attempts to weaken the detection or shut it off [15].

After the detection of an attack, another appealing functionality in IDSs is the response or counter measures deployment. A system can have a passive or an active response and they should not exclude each other.

In a passive response it is important that all the information about the detection is well logged so further investigation about this system abuse can be done. If some extra details from the attacked system can be captured, a deeper forensic analysis can be provided. Another passive response are the notifications of alarms or alerts to the network administrator, as well as SNMP traps.

The active response consists in reconfiguring other systems, like firewalls or routers, in order to block or mitigate the attack.

In my opinion, the active response is the most effective way of countermeasure, since it can block, minimize or restrict the attack. Furthermore, active responses need to be carefully and wisely used since IDSs can generate false positives.

But even any good system such as the ones described above cannot directly detect attacks within properly encrypted traffic. That is the reason why this work will focus its effort in developing an IDS that actually can detect attacks even when malicious data is encrypted.

2.5 Related Work

There are already some proposals in the literature that address the new methodology developed in this work. Although similar methodologies were proposed, none of them describes in detail the whole system architecture. When the system architecture is described, its description is too vague [21-23]. Moreover, the results shown are not complete, omitting important values, such as the number of *false positives* [24]. Another fact is that no work related to a complete system was found, only descriptions about how to set the ANN to detect illicit traffic.

Other papers exist showing a different approach to neural networks use: instead of being used to analyze traffic patterns, they are used to analyze log files [25]. Another method used was to locate keywords in the generated traffic and process them at the ANN [26]. Although the results of this work were good, the detection method could be easily overcame by the use of traffic encryptation methods.

2.6 Summary

This chapter presented the background of the dissertation. A small vision about how artificial neural networks are formed, how they work and their applicability was presented. Malware, especially botnets, have been discussed, focusing on some of the most interesting issues about them. To finalize, a brief view about how an IDS works and how it can help detect and prevent system abuses was examined.

After this introduction, we hope that the elucidation of how vulnerable a system can be and how exposed sensitive and confidential data are, can help to understand the importance that this issue really has, affecting private companies and governmental entities, that therefore must unify their efforts to at least minimize this problem.

3 DEVELOPED SYSTEM

In this chapter we are going to describe all the implemented functionalities, the reasons why these functionalities were added and the issues that were found during the development of our NIDS.

At a first stage, we will describe the general architecture, starting with its layers and a global and high-level overview. Then, we will describe the details of the proposed system and in particular their different modules.

As our NIDS was initially thought to be a botnet intrusion detection system, the software was named BotNet Security System (BoNeSSy).

3.1 Architecture overview

As the architecture of every system is one of the most important points and the key for a well designed system, some time was spent with this issue, evaluating the following main goals: versatility, reliability, usability, maintainability, cost, competitiveness performance, availability, security, integrability, functionality and portability [27-29].

After identifying all the desirable software characteristics, we have divided our software into layers in order to easily identify how we could design a versatile NIDS that could have its functionalities extended by simply adding extra plug-ins.

There are 2 main layers present at BoNeSSy: the development platform part, and the added value part. By the development platform we mean the base system with its hardware, operating system, database and core functionalities. The added value part includes all the extra functionalities that were developed on top of the base part: the monitor (the GUI) and the extra functionalities that are provided to the system, known as adaptation (plug-ins or add-ons), in order to support new threads. By dividing the system into these layers, we can assure maximum maintainability, integrability to new developed add-ons, and versatility. The next image (figure 4) shows the diagram of the architectural layer.

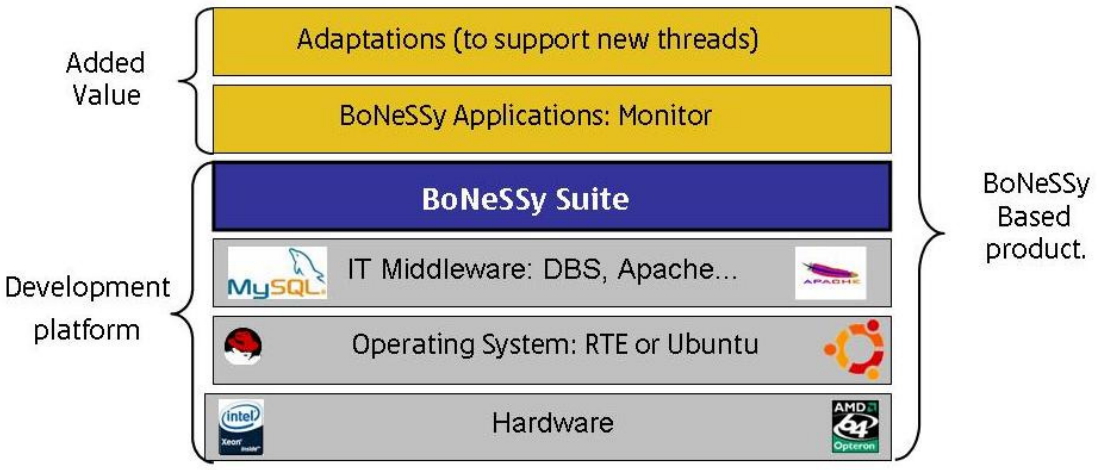

Figure 4 BoNeSSy architectural layer diagram

Having all these concepts in mind, we had to choose some of the third party software to provide functionalities and a stable base for our system.

Linux was the chosen operating system, because it is open, free and stable. The chosen distribution was *Ubuntu*, but *RHEL* or Centos OS would also be a very good choice. The next step was to choose the database. In our case, for development and testing in a small environment, MySQL is the most suitable technology, since it is lighter then *Oracle 11g R2*. However, this last option would be the chosen if BoNeSSy would run in a huge company's intranet due to the big amount of data that would be gathered. The GUI will be accessible from a browser, to ensure that the system will be available from any computer without any additional software installation. For this purpose, *Apache* was chosen to implement it.

The next image (figure 5) shows a big picture of the overall system, before any further explanation, so it can be easy to understand its general functioning.

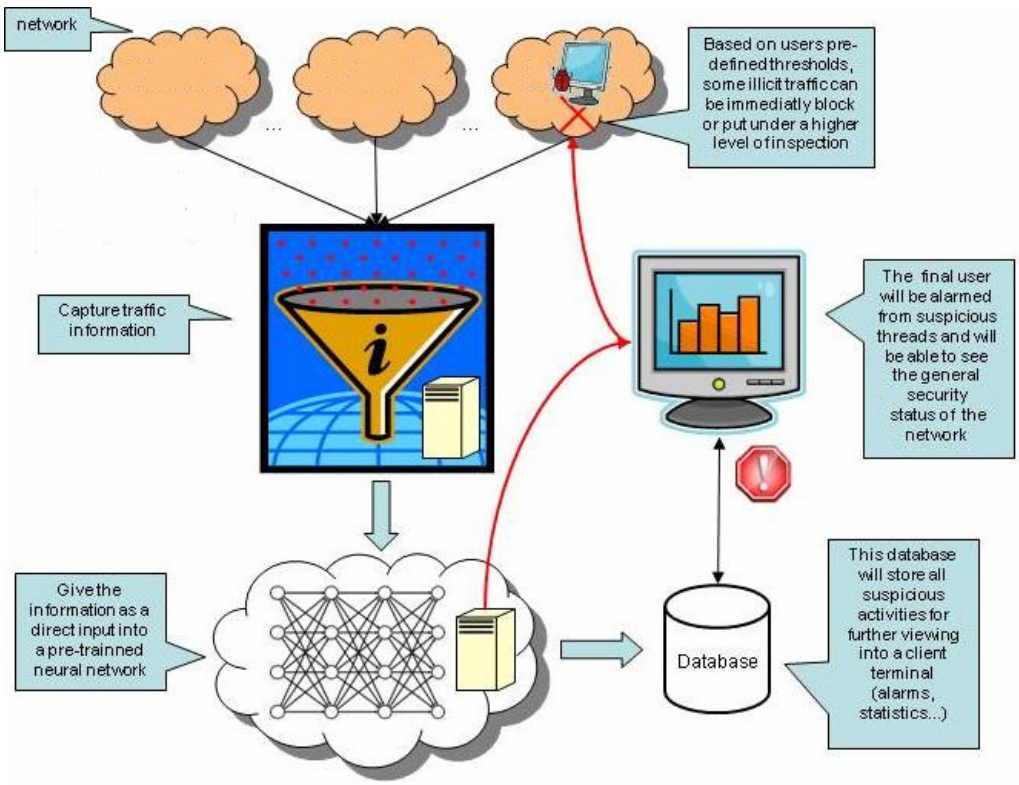

Figure 5 BoNeSSy big picture

A probe will be placed in key points of the network, so all the traffic data will be captured and treated to serve as input to the artificial neural network. The ANN will be placed at the server, together with the database and the HTTP daemon. This configuration is the stand-alone version; a distributed configuration can be easily achieved due to the fact that each part is independent from each other. Another feasible scenario could have the database with the HTTP daemon in a single host, the ANN in another and the data treatment (aggregations) in another one.

Given this brief overview of the general work flow of BoNeSSy, the system global architecture is shown in figure 6.

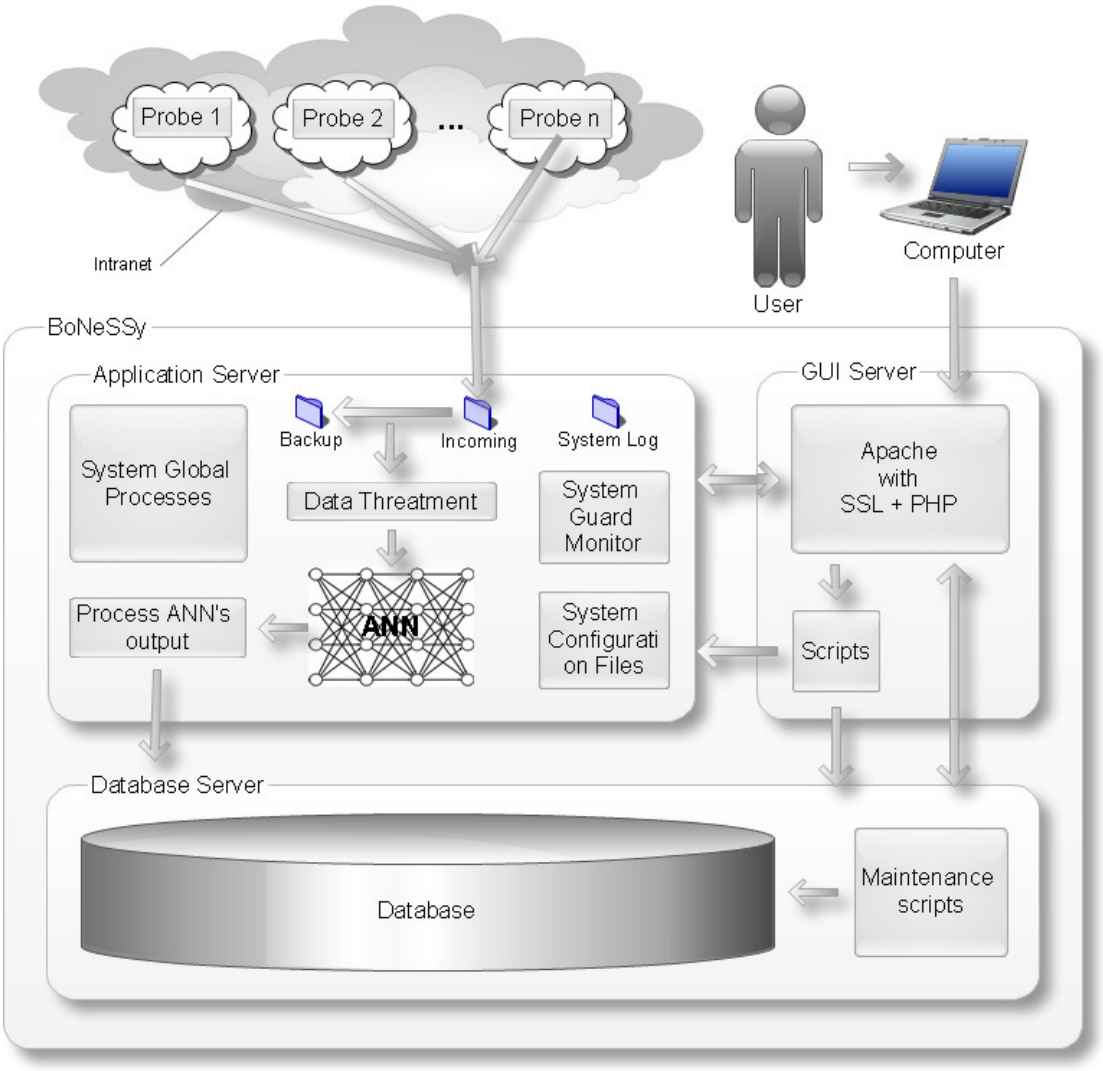

Figure 6 System global architecture

We want to make a clear distinction between the main modules of our application: by doing this, we are leaving the door open for a future split of each module between different servers in order to enhance performance and availability. For now, we are going to develop all these parts together, but always having in mind that each module can be distributed. Furthermore, the integration of the different modules with scripting will make it easier to provide both scenarios (distributed or stand-alone).

Probes

These are small programs written in C that will gather all network packets. These packets will then be forwarded to the Application Server through *SCP* in order to be

further inputted to the Neural Network. These probes will be placed in key points all over the network. Its configuration will be made by a configuration file.

Application Server

It's the core of the application, where all the data will be treated, processed and backed up as raw data. The output result of the ANN will be sent to the Database Server. At the Application Server, some global processes will also be running to maintain a consistent and healthy system. To finalize, System Guard Monitor will prevent that system functionalities stop working by monitoring them and making sure that, in case of a process failure, this process will run again automatically.

Database Server

As the name suggests, this component is responsible for all data storage, working as a *Data Ware Housing*. This server will also contain processing capabilities and will automatically trigger the statistical treatment of some parts of the data, so the results can clearly be displayed at the GUI Server.

Graphical User Interface Server

This module handles all user interactions with the software, providing a clean and easy to use interface where the user can monitor its network status, generate reports and manage the BoNeSSy suite and system global healthy.

For a better understanding about how the system works, how it was developed and which were the main issues, we are going to split the development phases into five main topics:

- Traffic Capture (probes)
- Application Server (Neural Network)
- Database Server
- Graphical User Interface
- Integration

3.2 Traffic Capture (probes)

Capturing the network traffic is the start of our software work-flow. It is extremely important to have an efficient way to capture and filter the network data. This probe should

be configurable and fast enough to identify and capture the network traffic that the user wants to monitor. For this purpose, a small program written in C was developed. This program works in promiscuous mode, capturing all traffic that passes at the network despite of having the packets addressed to it or not. The name of this program (probe) is *Trafana Lite*. This name came initially from a program that was developed by Professor Salvador named *Trafana*. This program had many functionalities and extras. But as this program had many *if* conditions and extra variables, we decided to develop a new one with less functions, a lighter brother, so it would consume less system resources and be faster.

For developing this program, we used the *libpcap* library. The syntax to use this program is:

```
./trafana-lite device_number sample_time total_time ip_file
```

where the device number is the number returned by the *pcap_findalldevs* function, *sample_time* is the time during which it captures packets before writing a line to the file, the *total_time* is the total time the program should run and the *ip_file* is a file that has all the IP address whose traffic should be captured.

As inputs to the capture process, we have the device number, that can be shown with the option "-l", the period of time for each sample, the total period of time for the whole process and a file containing the IP addresses and the network masks of the packets that are going to be captured. The file syntax should be like this:

```
[IP Adress] [Network Mask]
[IP Adress] [Network Mask]
…
```

To update this file, we had developed a small script that fetches new entries from the database regarding the specific probe and overwrite the configuration file. This script is called by *crontab*. Since running this script is optional, the user may have to configure all system topology manually. But if this script is activated, the configuration is centralized at the GUI. The script's name is *bonessy_topology.sh*

As output, this program writes multiples files, each one corresponding to a single IP address. In this way, it will be easier to store, identify and differentiate the different inputs of the ANN. The output files are named according to the captured IP address, like for example:

```
ip-10_46_17_152_YYYYMMDDhhmmss.log
```

where 10_46_17_152 is the IP address, the *YYYY* is the year, *MM* the month, *DD* the day of the month, *hh* the hour (in a 24 hour format), *mm* the minutes and *ss* the seconds. Inside of these files, we have stored 4 values in four different rows, divided by a space:

- Number of input packets;
- Number of output packets;
- Size of input packets;
- Size of output packets.

Each row corresponds to the data captured on the period of time of each sample that is specified as an input parameter. These values are stored because the ANN will receive them as input, since this was the method that was used for illicit traffic detection in previous studies [30].

For a better understating of *Trafana Lite*, the program work-flow is shown at figure 7.

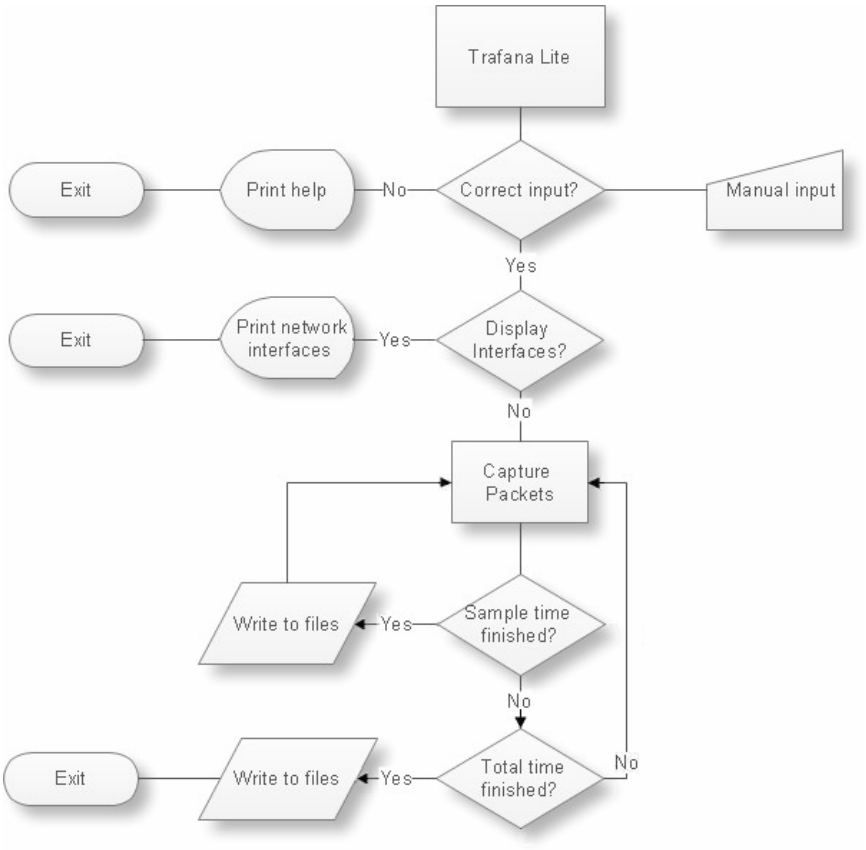

Figure 7 Trafana Lite work-flow

This program will be placed in several key points at the intranet, so all related traffic from the targets will be captured.

3.3 Application Server

The application server is the core of our program and will hold all the key operations of the system. Since this is the most important part of BoNeSSy, its description will be exhaustively discussed in four sub-topics:

- Data Treatment;
- Artificial Neural Network;
- System Global Processes;
- System Guard Monitor

Data Treatment

Before the data can serve as input to the ANN, it is mandatory to have some transformation, in its syntax and content.

The first step is to normalize the data, as it comes from different devices with different network throughputs [31]. For this purpose, a small program was developed. This program receives as input the file with all the data that was captured by the probes, set its mean to zero, its standard deviation to 1 and outputs a new file. The formulas for the mean and standard deviation can be seen in equations 1 and 2, respectively:

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \quad (1)$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N} (x_i^n - \bar{x}_i)^2 \quad (2)$$

The normalization of the data is very important, since it can influence the ANN output. The formula for normalization is the following:

$$\bar{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i} \quad (3)$$

After data normalization, there is another important data treatment phase that can improve ANN's performance and minimize its processing time: data aggregation. Data

aggregation should be carefully used since small anomalies can be hidden if the level of aggregation is too high.

For this purpose, another small program was developed, that receives as parameters an input file, an output file and its level of aggregation. After this, the program sums the number of lines based on the level of aggregation and outputs its sum to another file.

In order to have an appropriate data syntax format for the ANN, another small program was developed. This program formats the input file according to FANN parsing rules with an informative header. It uses a sliding window that is represented by a vector: for example, if a vector v_0 has [x…y] elements, the next sliding window will be represented by a vector v_1 that will have [x+1…y+1] elements, according to the formula v_n [x+n…y+n]. This program receives the following inputs:

Input file;

Output file;

h value (input data vector size);

The data you want to filter:

1 – packets in, size in;

2 – packets out, size out;

3 – packets in, packets out;

4 – size in, size out.

Output value: for generating guided training file.

By using these three simple programs, the data will be ready to serve as input at the ANN. The work-flow of the data treatment phase can be seen at figure 8.

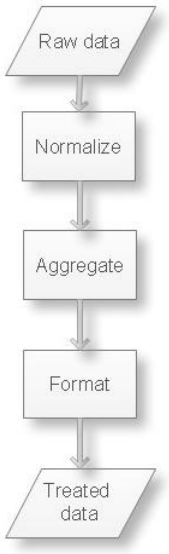

Figure 8 Data treatment work-flow

Artificial Neural Network

To develop our ANN, a research was made about which neural network types where better for our problem to identify network traffic patterns. Although there are several types and algorithms for ANNs, one in special is more suitable for non linear pattern matching. We will use a feed-forward network model using the back propagation learning algorithm. Back propagation is a general purpose learning algorithm for training multilayer feed-forward networks that is powerful but expensive in terms of computational requirements for training [30]. We will try to use a supervised training with the *Levenberg-Marquardt* training algorithm with automated Bayesian regularization that is proved to be effective in detecting illicit traffic patterns [30].

Having this in mind, we start looking for an open ANN library with this algorithm, since developing an ANN was out of the scope of this work. During our search, there was only one open source library that had the *Levenberg-Marquardt* training algorithm: "ANN++ - Artificial Neural Networks in C++" [32].

Besides being currently abandoned and inactive since 2002, this project was the one that best matched our criteria. Before downloading the library, a warning was displayed: "A pre-release version of ANN++ is available for download. Please note that this is an unsupported and undocumented release. It does work though."

These were some disappointing news; nevertheless we downloaded it and tried to compile it. During the attempt to compile the library, a lot of issues pop out, from wrong function definitions to bad linking. Moreover, no documentation was provided to help us.

A new search for a more documented and easy to use ANN was required due to failure into putting this library to work. So we decided to use the FANN [33].

After installing and reading the documentation, we developed three programs, where each one is able to handle different input data vectors' (*h*) sizes: five, ten and fifteen. Each *h* value is constituted by a pair of numbers, that can be a mix between traffic in, traffic out, packets in and packets out. These programs were developed for testing purposes. In this way, we are able to create, train and run the ANN all-in-one. The programs have the following inputs:

- Algorithm (these algorithms were already explained in the Introduction chapter):
  - FANN_TRAIN_INCREMENTAL;
  - FANN_TRAIN_BATCH
  - FANN_TRAIN_RPROP
  - FANN_TRAIN_QUICKPROP
- File to train
- File to test
- Log file

With these programs, the ANN testing and calibration phase will be easier. As a default ANN configuration we set the following values:

- Number of layers: 3
- Number of hidden neurons: *h**2
- Number of input neurons: *h**2
- Number of output neurons: 1
- Desired error: 0.0001
- Epochs between reports: 10
- Max epochs: 3000

We used 3 neuron layers because according to the theory of Kolmogorov, a Russian mathematician [34], a 3-layer BackPropagation (BP) model can express a mapping from *n* dimension to *m* dimension, that is to say, for a general application one hidden layer is enough. Thus, we set one hidden layer in our program.

Note that these values are not optimized. It is just a starting point to verify how accurate the ANN is. But, while testing, we will always have in mind the importance of the number of hidden neurons and epochs to avoid problems like ANN overfitting. This topic will be further developed during the test phase.

These programs could have an extra input parameter to switch from creating the NN, to training the NN or testing the data. Therefore, we thought it would be better to have a

program with hardcoded best settings that would be used at the NIDS to analyze all the data. Hence, we also developed a NN program that only tests the data for illicit traffic patterns based on a pre-trained NN.

The work-flow of the *nn_trainh5, nn_trainh10* and *nn_trainh15* programs described above (training plus testing) can be seen at figure 9.

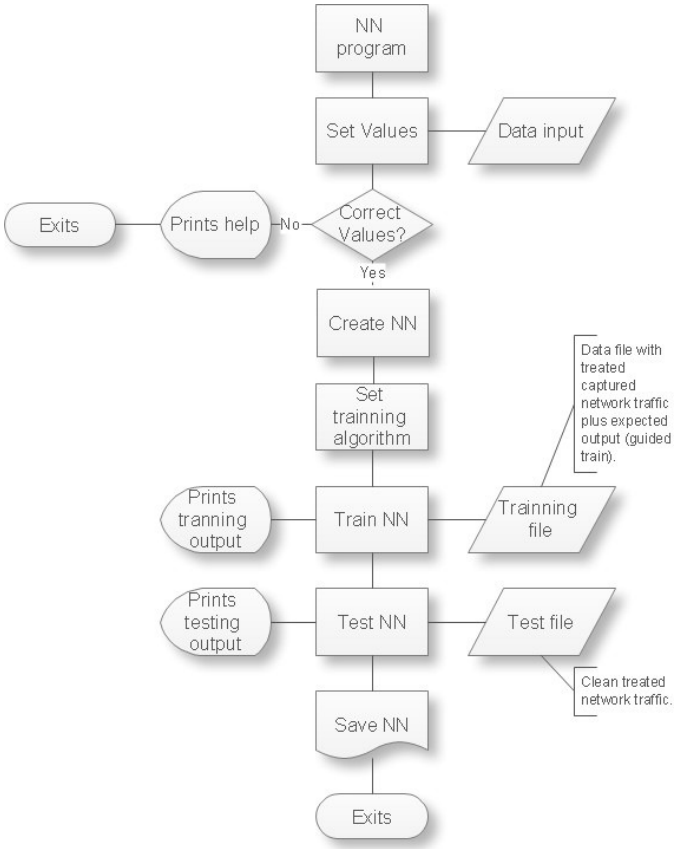

Figure 9 Work-flow of the nn_trainh5, nn_trainh10 and nn_trainh15 programs

After developing a program for testing purposes, we developed a simpler program that only loads a pre-created ANN, with the help of the previous programs, and tests a given data file for illicit traffic patterns. The name of the program is *bonessy_ann_test* (figure 10).

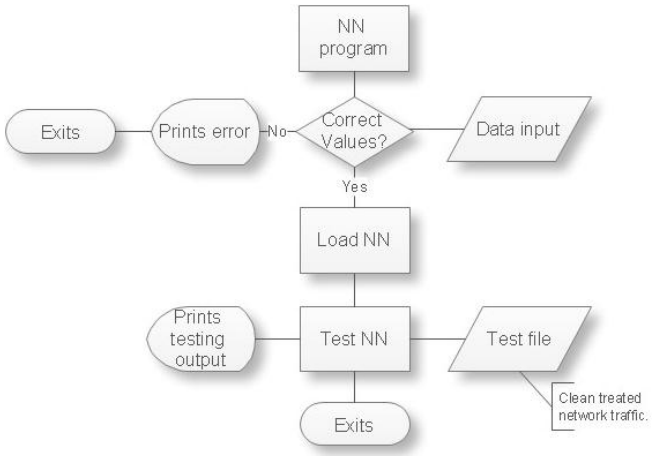

Figure 10 Bonessy_ann_test work-flow

In order to add value to a previously saved default ANN, we have also developed a program that only trains the ANN. This program is important for adding extra capabilities to any ANN. The workflow of this program (figure 11) is almost the same of the previous one, but instead of receiving a set of data and running it, it receives a training set and enhances the ANN used in the above program. This program was named *bonessy_ann_train.*

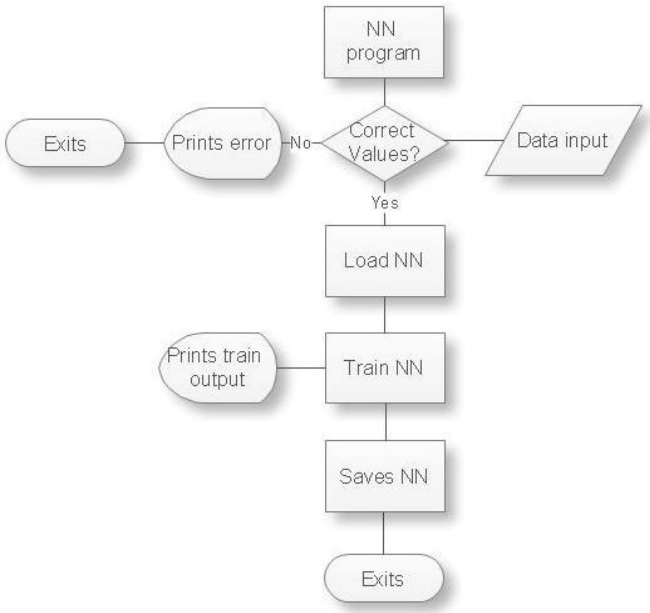

Figure 11 Bonessy_ann_train work-flow

In this chapter, only the functional code was delivered; no ANN performance and accuracy tests were made. As previously said, this topic will be developed during the system testing phase.

3.4 DataBase

This sub-chapter will deal with all system database creation, from choosing the database provider to effectively creating it. As discussed before, for testing and small environment utilizations the MySQL technology is the more appropriate one to our system. It is free, easy to use, easy to backup and lite. The only problem is the database scalability, because it cannot be run in distributed servers in order to enhance its performance. Although oracle is able to be upgraded to work in cluster environments (*Oracle RAC*), MySQL is not. Another good choice to obtain good performances with huge amounts of data, much better than *Oracle RAC*, is *GreenPlum*. This new database has new concepts and a new architecture that in most of the cases can go faster than Oracle RAC, but these distributed database environments are for huge systems that would be able to handle all the company data, besides being extremely expensive. Because of those reasons, it is out of the scope of this project.

A program named DB Designer 4 version 4.0.5.6 was used to design the database. With this program we were able to graphically design all the database tables, primary keys and foreign keys. This program also gives the functionality to export these contents to a SQL file that contains all the definitions, so we can simply run this file inside the MySQL program through the command line, and easily create our database.

At figure 12 it is possible to view the designed database scheme.

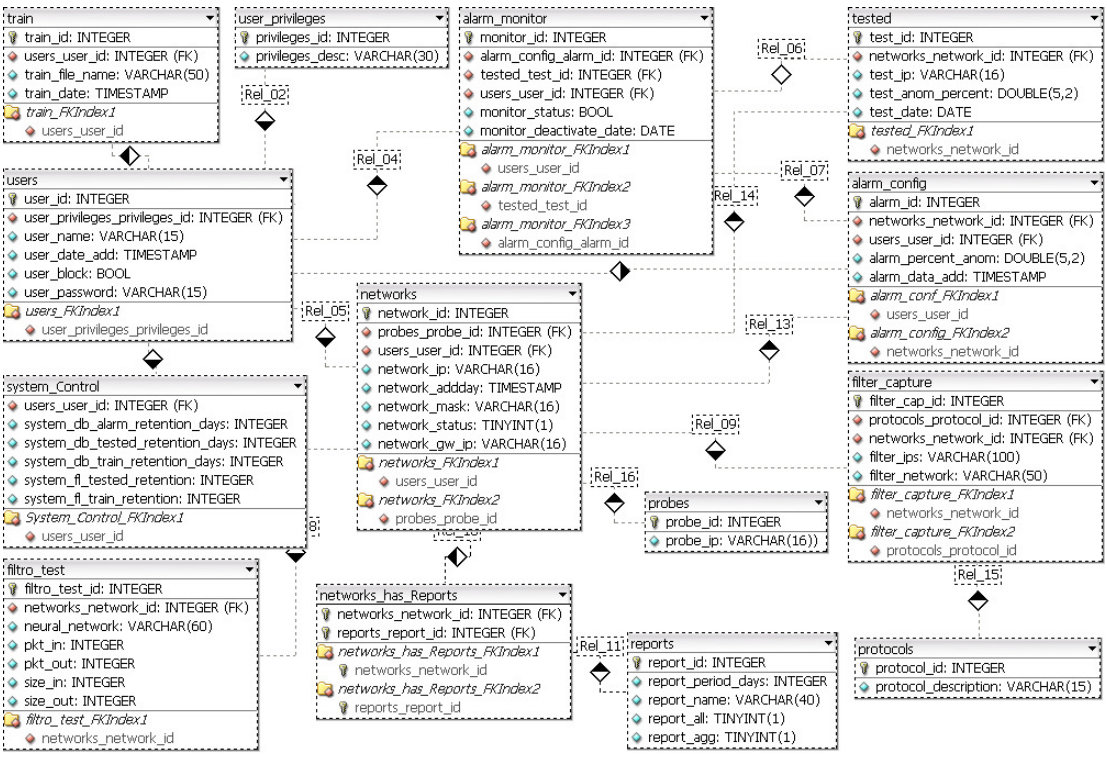

Figure 12 The BoNeSSy database schema

In order to fulfill the needs of BoNeSSy, we created fourteen tables that are responsible to handle all the user defined parameters and all the network alarms statistics. The symbol ◇ means a 1:1 relation between the tables, the symbol ⬖ means a n:1 relation between the tables and the symbol ⬗ means a 1:n relation between the tables. For a better understanding, a brief description of all tables will be provided. In order to avoid this text from being repetitive, we will omit the *id* columns from all the tables:

- Train: This table has stored all the training files definitions that the user wants to upload to the system. It associates a training file (train_file_name) with a user (users_user_id) and that's why we have a foreign index and a data (train_date). These extra columns are for logging and security purposes.
- Users: This table is responsible for storing all application users for system authentication purposes. All users have an unique id (user_id), one or more associated privileges (user_privileges_privileges_id), the date of creation (user_date_add), its actual status (user_block), which is 0 if it is operational, 1 if blocked and 2 if suspended, and a password (user_password) that is cryptographed.

- User_privileges: It contains for now, two possible privileges: 1 – administrator and 2 – guest. The administrator will be able to set and update system parameters, as well as to create other users, backup the system and restore the system, among other tasks. The guest will only be able to check the system status and run reports.
- System_control: This table is responsible to store some system definitions, its alarm retention at the database (system_db_alarm_retention_days), that is, the time interval that will be used to save network alarms (0 – infinite time, X – for X days), the time interval in days that will be used to keep the tested data in the database (system_db_tested_retention_days), all the ANN training records in days (system_db_train_retention_days), all the files that were captured by the probes and are used as inputs to the ANN (system_fl_tested_retention) and all the system training files that were uploaded to the system (system_fl_train_retention).
- Alarm_monitor: This table is where all the alarms are stored. Whenever a ANN output is higher than expected a record is inserted in this table pointing to the tested data (tested_test_id), to the alarm threshold configuration (alarm_config_alarm_id) and to the user that disabled the alarm (users_user_id), changing the monitor status (monitor_status) from 0 (enable) to 1 (disabled). The date where the alarm was disabled (monitor_deactivate_date) is also logged. This table is filled in by a trigger named alarm_trigger and every time a new entry is added to the table the trigger checks the user configured thresholds - if the anomalies are higher than the user input, a new row is added.
- Tested: All the outputs from the ANN will be logged here, so one entry will be created every time the ANN analyses an incoming data from the probe. For every entry, a network is associated to the data (networks_network_id), its corresponding IP address (test_ip) (since each data file is related to one IP), the percentage of anomalies (test_anom_percent) and the date when the test was run (test_date).
- Alarm_config: This is the table where alarms are configured and any user with appropriate permission can add an entry. It defines when the alarm should be prompted using a threshold (alarm_percent_anom) that is associated with a network (networks_network_id). By adding a new entry, the user credentials (users_user_id) and the alarm data (alarm_data_add) are logged.
- Filtro_captura: This is related to the probes, where we can define which IPs to filter (filtro_ips), if any, or in a higher level, the network to filter (filtro_network). All this

data is associated with a network (networks_network_id) and protocol (protocols_protocol_id) to apply to the selected network or IP.

- Protocols: This table only lists the protocols ID's (protocol_id) and descriptions (protocol_description).
- Networks: Here, all the networks that the system is supposed to monitor are stored. Each network (network_ip) and mask (network_mask) is added by a user (users_user_id) in a certain date (network_addday). It has also a column to indicate its status (network_status), in case it is in a special state, like audit mode (depper trace) or blocked (the router was reconfigured to block the network or some specific IP). There is also a column that associates the network with the router or bridge (network_gw_ip), so the system will know where to send the commands to block any misbehavior IP address. This network table also supports a single IP entry as the network.
- Probes: All the probes responsible to collect data from each network are listed in this table. Each probe is uniquely identified by its *probe_id* and its IP address (probe_ip). Each probe can capture one or more networks at the same time (networks.probes_probe_id).
- Network_has_Reports: As the tables of the networks and reports have a n:m relationship, this table must be added to control this relation. So, it logs the corresponding PK from each table for the sake of consistency.
- Reports: The reports definitions are stored here. The periods of the days (report_period_days), its name (report_name) and if it is a global report with all the networks or not (report_all) are shown here. It is also possible to define the level of aggregation (report_agg) in days (1), weeks (2) or months.

After creating this scheme, this data was exported as "sql creation script". Now, it is just a matter of running this script at MySQL database to have all these tables ready to receive data.

At this point, all database modifications are logged, the users that changed the settings and the date of the modifications are also saved.

3.5 Graphical User Interface

In order to facilitate the system configuration and monitoring, we developed a user-friendly interface based on PHP and CSS. For system security, we will use apache and SSL. With this interface, the user will be able to:

- Securely login into the system (SSL);
- Check system overall status;
- See system alarms;
- Monitor the server;
- Check the status of the processes and start/stop them;
- Configure reports;
- Generate reports;
- Add new networks to monitor;
- Configure new alarms;
- Train the ANN with new features;
- Add probe filters;
- Change system paths;
- Configure DB and files retention;
- Add, remove, lock or unlock new users;
- Backup and restore the system;
- Re-install the system;
- Free disk space;
- Read documentation.

To access the login page, just type: https://<server_ip>/login.php (figure 13).

Figure 13 BoNeSSy login page

This login page (figure 13) uses the md5 function from PHP to encrypt and decrypt the passwords from the database. The approach that was followed is the one described at [35].

After the login, the user is redirected to the system starting page (figure 14).

Figure 14 BoNeSSy starting page

At this page (figure 14) the user can access all system functionalities, if it has administration role. At all the time, the logout functionality is always available by pressing the logout button at the left bottom of the page.

If the user wants to check the system global status, it just needs to press the "Global Status" link at the main menu. After pressing it, the user is redirected to another page – the overall status page (figure 15).

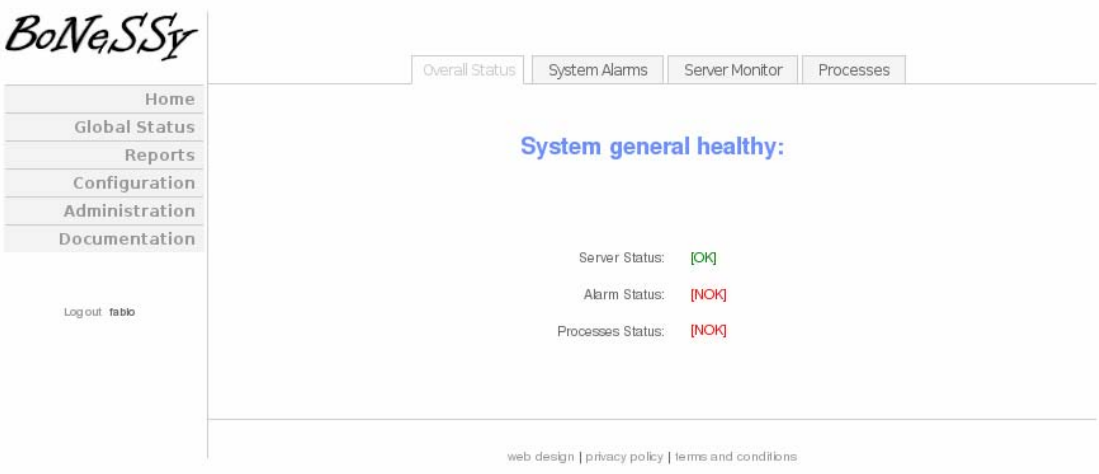

Figure 15 BoNeSSy overall status page

In the above page, it is possible to quickly verify the system general healthy by checking the status: [NOK] will signalize the user that there are some issues that need to be solved and [OK] will signalize when there are no issues. The Server status is a summary of the server global healthy (memory, processor, disk), the Alarm status shows whether there is any alarm prompted or not and the Processes status shows if there is any process down or not.

It is possible to analyze the error by clicking in its corresponding sub-menu on top of the page.

For system alarms status, just click at “System Alarms” and you will be forwarded to the page that is shown in figure 16.

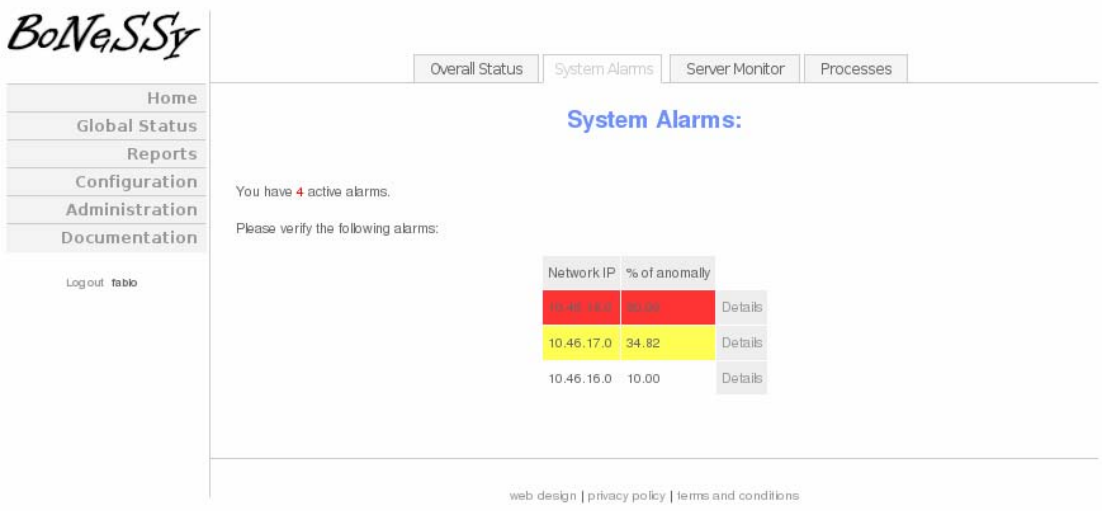

Figure 16 BoNeSSy system alarms page

At this page, the user can check from which networks the alarms came from, as well as its percentage of anomalies (figure 16). At this page, only the highest entrance for each network will appear.

For more detailed information about the alarms, it is possible to press the “Details” button. In our case, we are going to check the details for the first alarm (figure 17). At this page, all the triggered alarms will appear.

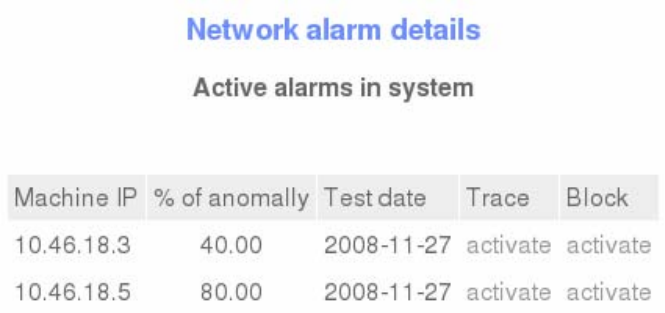

Figure 17 BoNeSSy network alarm page

From figure 17, it is visible that for network 10.46.18.0 two alarms were prompted. The higher alarm is the one that is displayed at the main system alarms page.

If the user wants to activate the trace mode utility for further analysis, it is also possible. This trace utility captures all the data from or to this IP address, regardless of the protocol. For drastic measurement it is possible to block the machine, setting the router to deny any request from that IP address.

The next page (figure 18), corresponding to the sub-menu “Server Monitor”, shows the server healthy status. This functionality is provided with the help of the *sysstat* package that has the *sar* command.

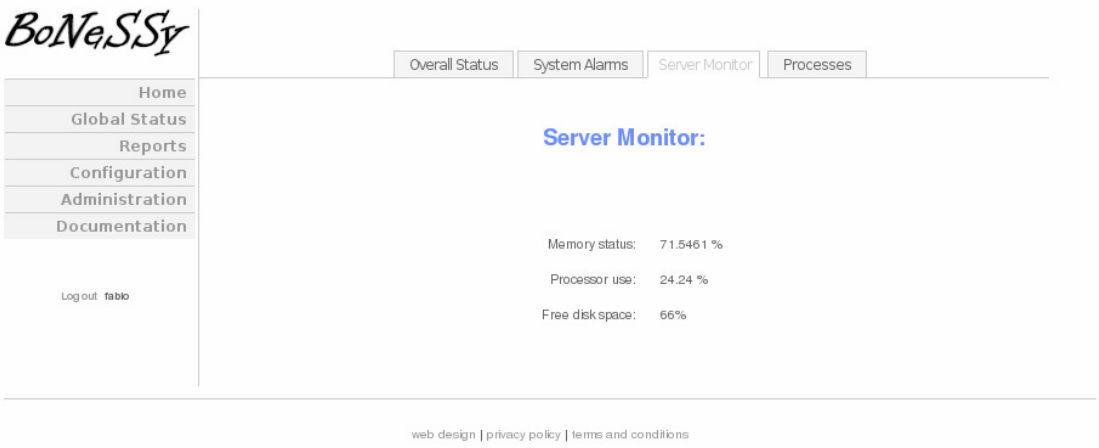

Figure 18 BoNeSSy server monitor page

If the server reaches critical levels of memory, processor or disk space, this information will be shown at the “Overall Status” page.

To monitor all main system processes, just click “Processes” at the sub-menu. At this page (figure 19) it is possible to check the processes’ status as well as start or stop them.

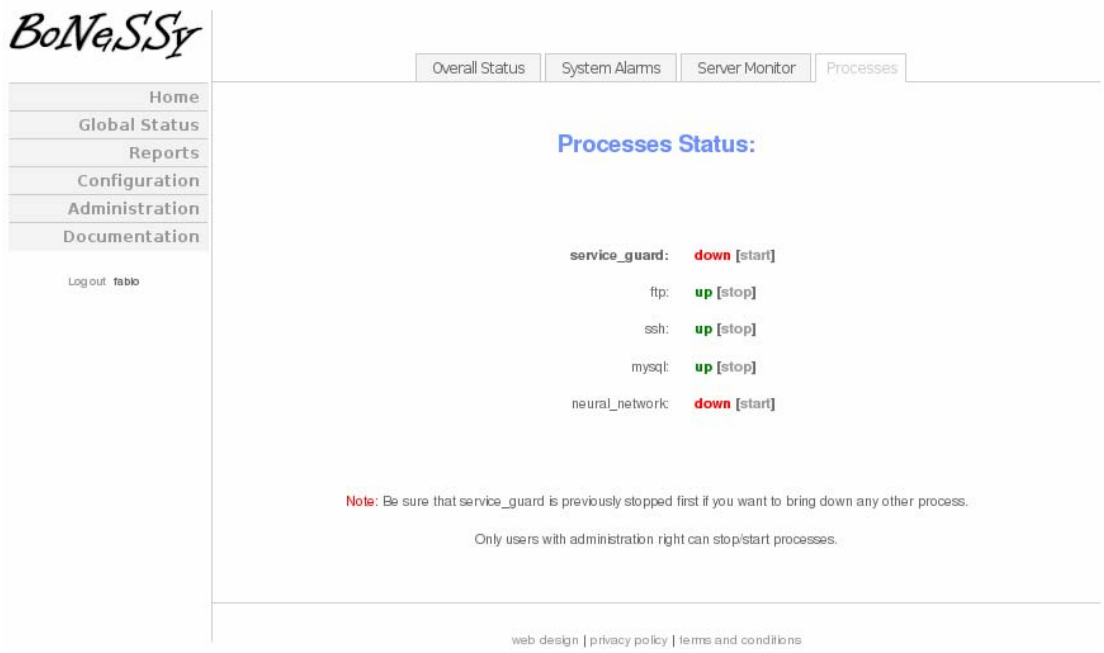

Figure 19 Status page of the BoNeSSy processes

To start a process, just press [start] and to stop press [stop]. In our example we have two processes down, the service_guard, which is responsible to monitor any program that was previously configured, and the neural_network. It is important to note that for these services to work we need to configure the sudores beforehand to include the following lines:

```
root  ALL=(ALL) ALL
www-data ALL=NOPASSWD: /bin/kill, /etc/init.d/service_guard,
/etc/init.d/httpd, /etc/init.d/ssh, /etc/init.d/mysql,
/etc/init.d/proftpd
```

After pressing the [start] button for *service_guard*, the window at figure 20 will be prompted:

Figure 20 BoNeSSy process starting window

After this, the service will be up and running. Note that only users with administrator role can start/stop processes.

This ends all the functionalities present at the “Global Status” section.

Now we will show the functionalities available from the “Report” section. This section can be accessed by pressing the “Reports” button at the left menu. When accessing this page, the user will be able to generate previously configured reports by pressing the “Generate Report” button as shown in figure 21:

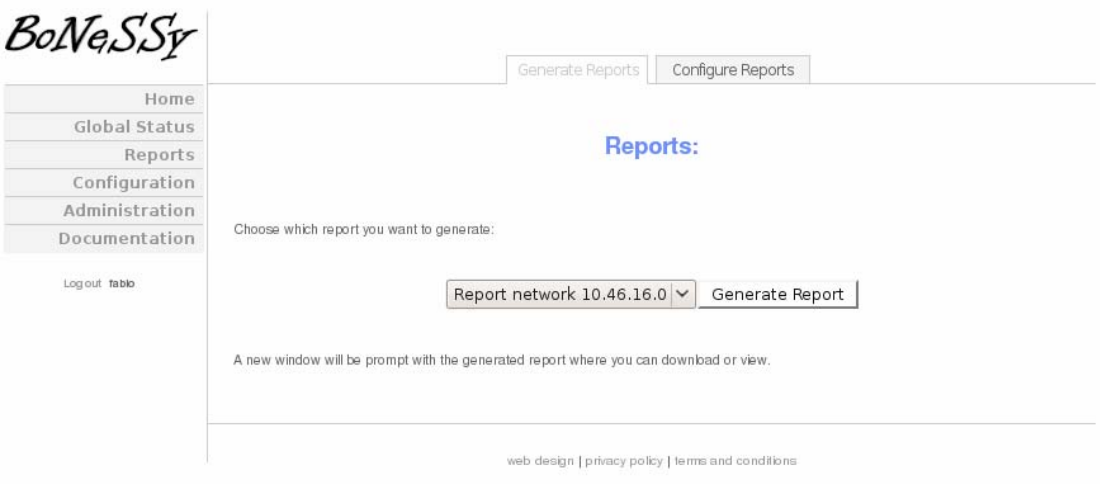

Figure 21 BoNeSSy generate report page

While the report is being created, a window will be popped-up as shown in figure 22.

Figure 22 BoNeSSy report generation window

When the report is ready, the window will be refreshed and the user has the option to view it directly through the browser or download it in *PDF* format (figure 23).

The report is created in HTML and converted to *PDF* with the help of the *htmldoc* utility. The graphics are generated by *octave*, so we need to convert from *.eps* to *.png* (the *convert* tool was used for this task).

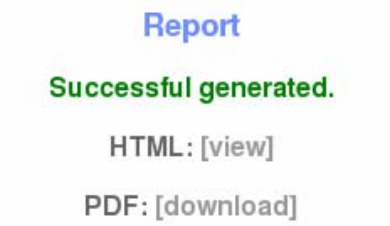

Figure 23 BoNeSSy successfully generated report window

The report will contain a graphic for the selected time period with the number of anomalies (alarms) for the specific network. The IP address with more alarms as well as the one with the highest tested percentage of anomalies will be displayed. An example of a report can be seen at figure 24.

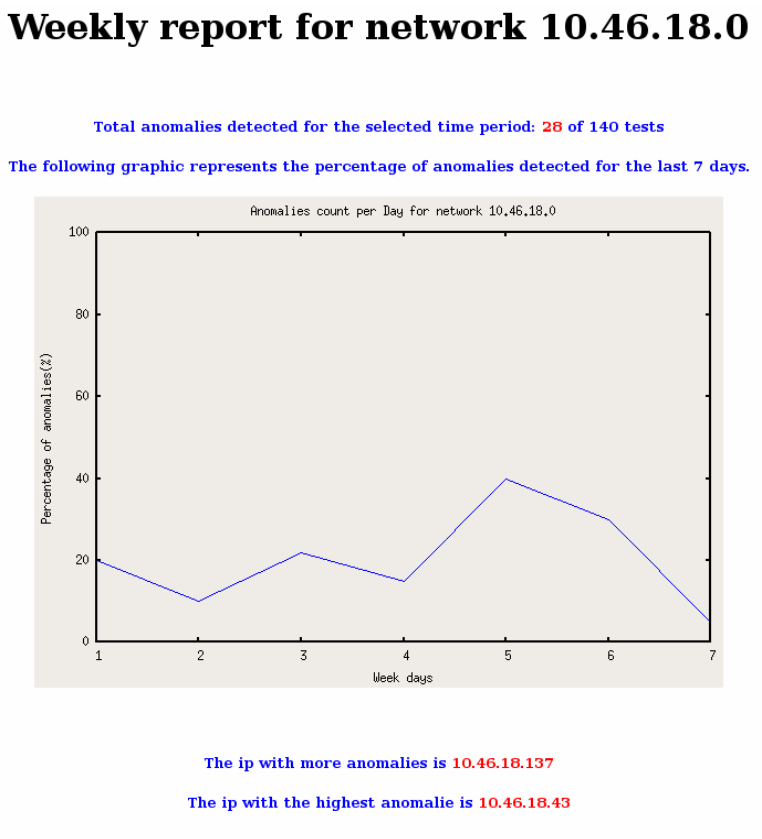

Figure 24 BoNeSSy report visualization page

In order to use the report that was shown in the previous page, firstly it is necessary to create it. At this page (figure 25) the user can create and delete reports.

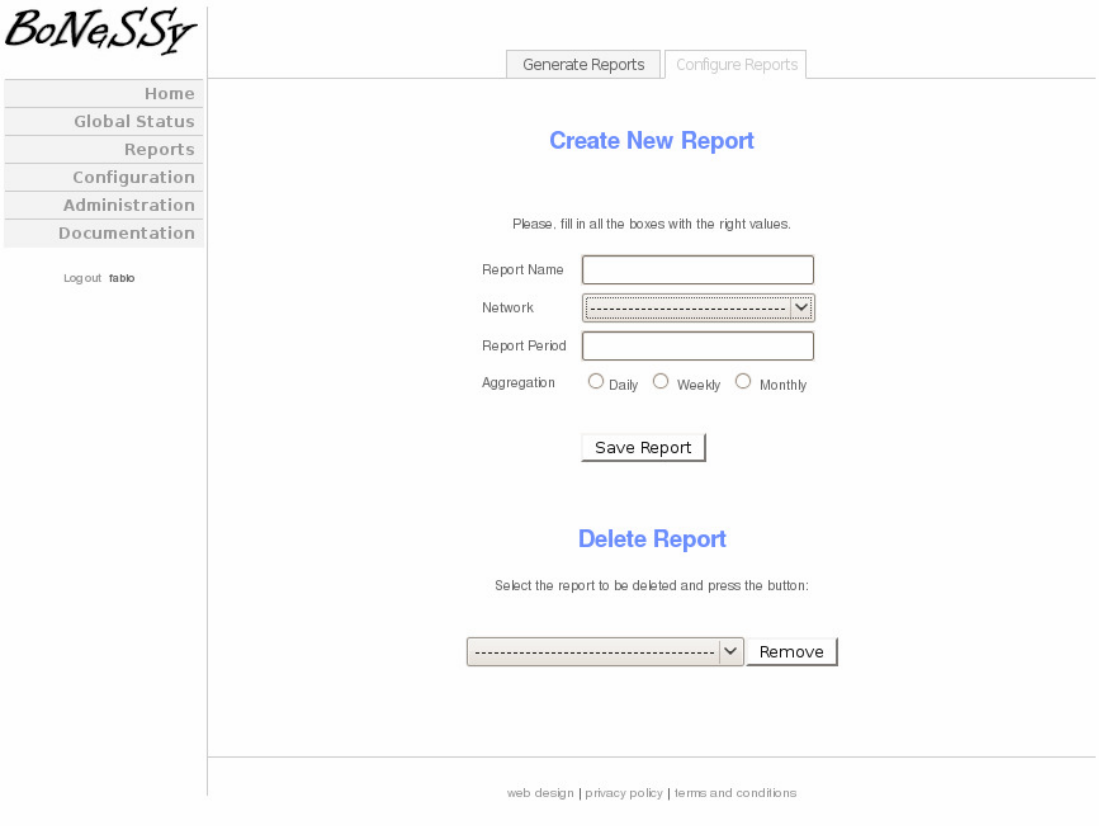

Figure 25 BoNeSSy report creation page

For system configuration, we have a section named “Configuration”. Here, the user can add/remove networks, alarms, filters, train the ANN and other options. When this section is selected, the user is forwarded to a page with a disclaimer, alerting that only users with administration role can access from there on (figure 26).

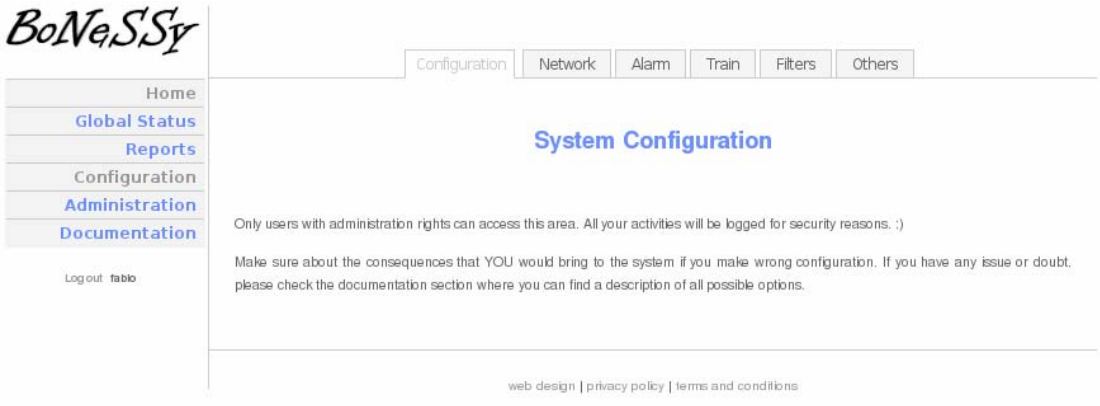

Figure 26 BoNeSSy configuration main page

Following the first option at the sub-menu, we will be redirected to the network configuration page. Here the user can add new networks to the system. This step is mandatory, because without a network in the system it won't be possible to create rules to capture specific traffic, so the probe won't capture anything. At this page (figure 27), it is possible to add and remove networks.

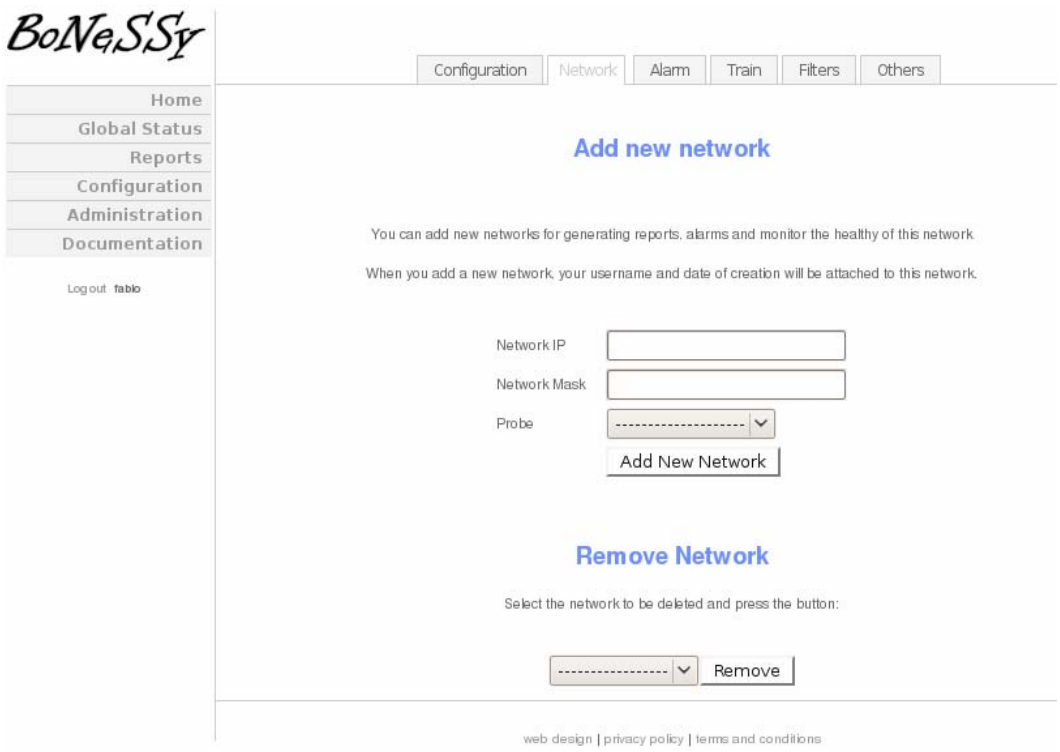

Figure 27 BoNeSSy add new network page

The next option is the alarm configuration page (figure 28), where it is possible to add new alarm thresholds or remove them. These alarms are the ones that can be followed by at *Global Status > System Alarms*.

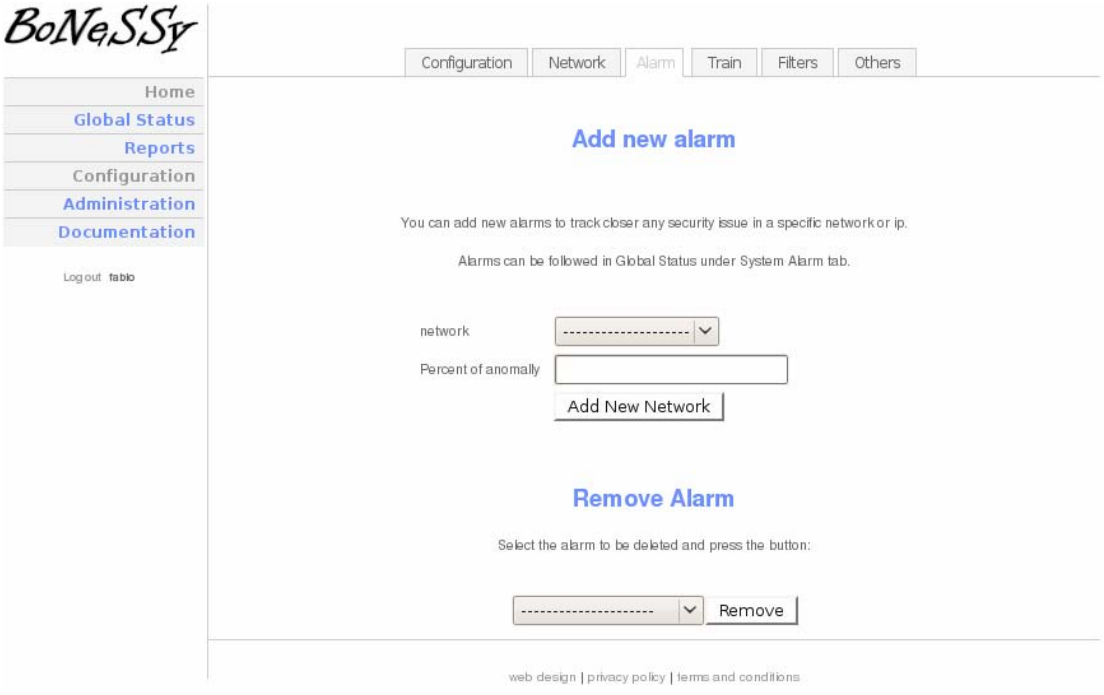

Figure 28 BoNeSSy add new alarm page

It is also possible to enhance the neural network by training it with extra data (figure 29), so it can detect other anomalies besides the default ones. For this, it is necessary to point to the location of the new training set, with the help of the "browser" button and by pressing "train". At this moment the train file will be located in a specific folder named "train", and runs the program *bonessy_ann_train* with the training file as input.

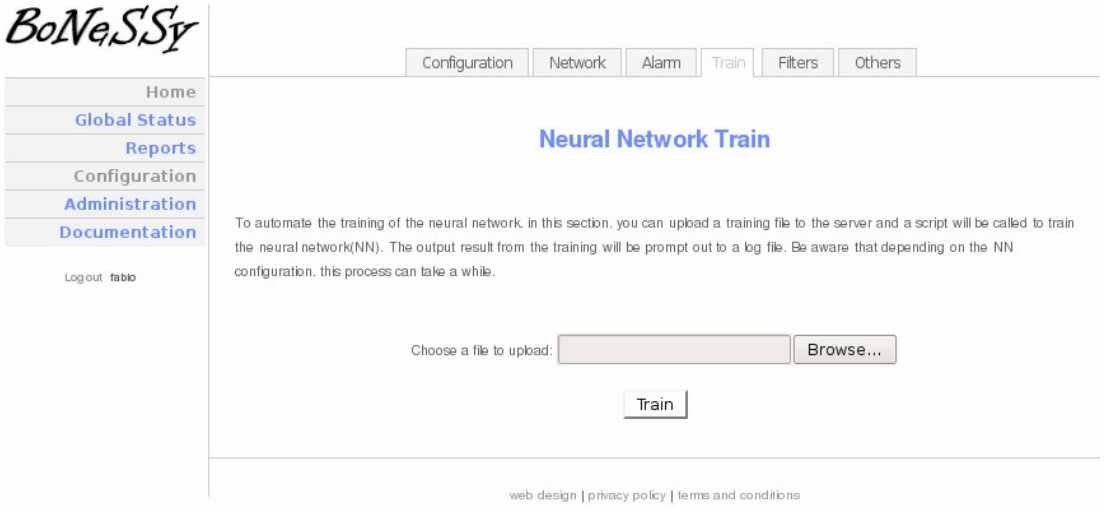

Figure 29 BoNeSSy train neural network page

It is also possible to add test filters to each network (figure 30). This means that it is possible to specify the current input for the neural network: packets in, packets out, size in and size out. This option has to be used extremely carefully since it has to respect the way the ANN was trained to receive the data. Nevertheless, this can give some flexibility to the system.

The possibility to add probe filters is also available through the interface (figure 30). This will save in the database the protocols the probe should capture (all, FTP, HTTP, SIC). Although this is not yet supported by the probe, it is already implemented at the GUI and database, so in the newer future it will be easier to develop.

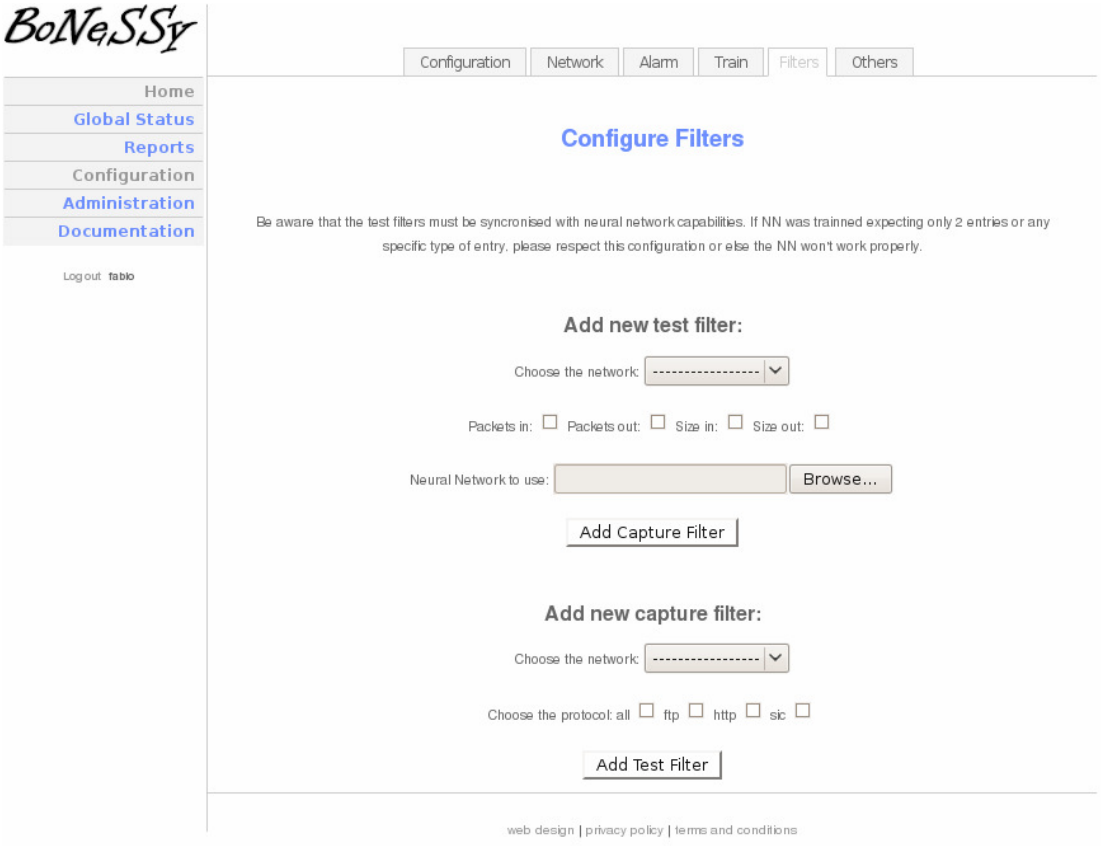

Figure 30 BoNeSSy add new filter page

The last page from the configuration section is named *Others* (figure 31). Here, it is possible to set some system global configurations like system paths, database and files retention times.

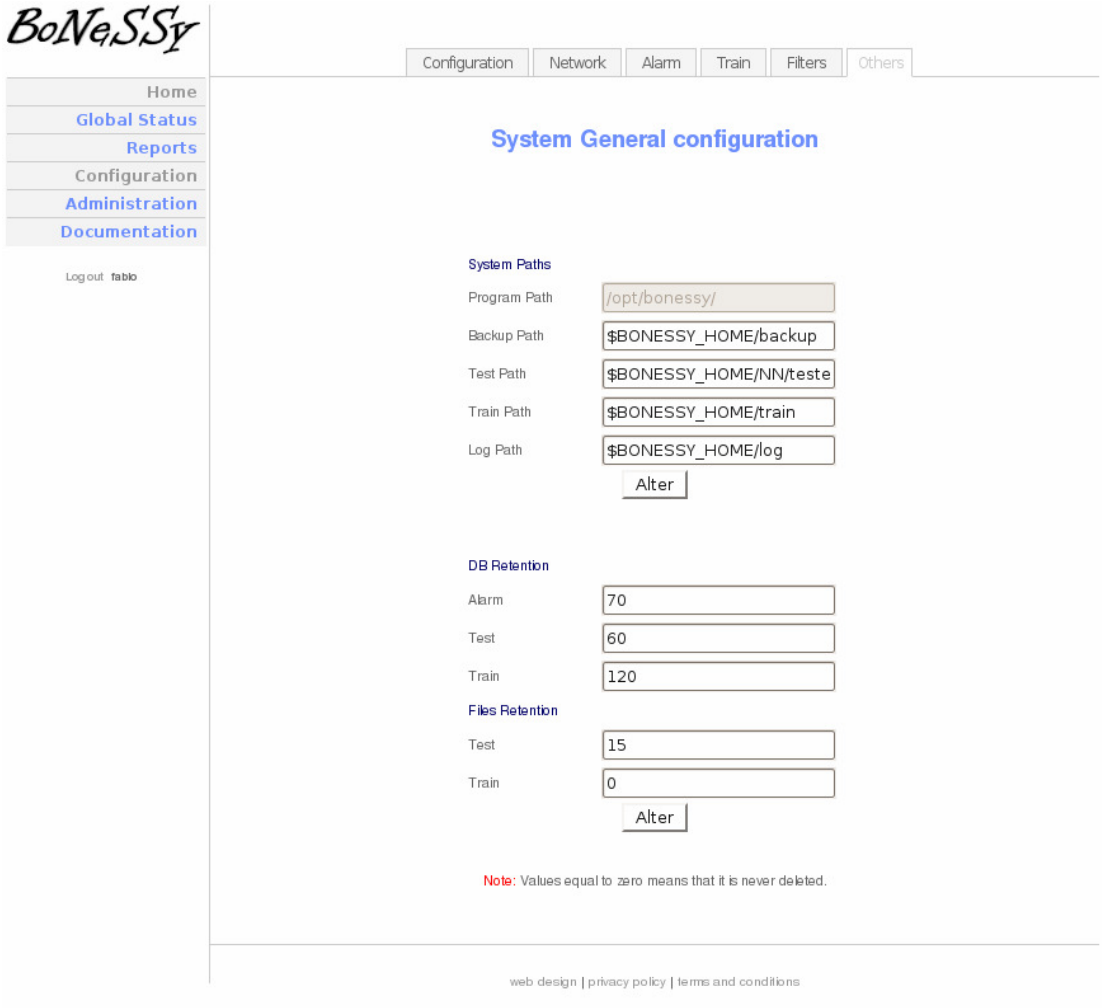

Figure 31 BoNeSSy general configuration page

As said at the bottom of the page (figure 31), if zero is set for the retention value it means that the files or the database information will never be deleted, which is not advisable in case of limited storage space.

To help system administration we have another section named “Administration” that is accessible through the left menu. As in the “Configuration” section, when clicking in “Administration” the user will be redirected to a disclaimer (figure 32) page that advertises him for the consequences that any wrong settings can bring to the system and that only users with administration role can access this functionality.

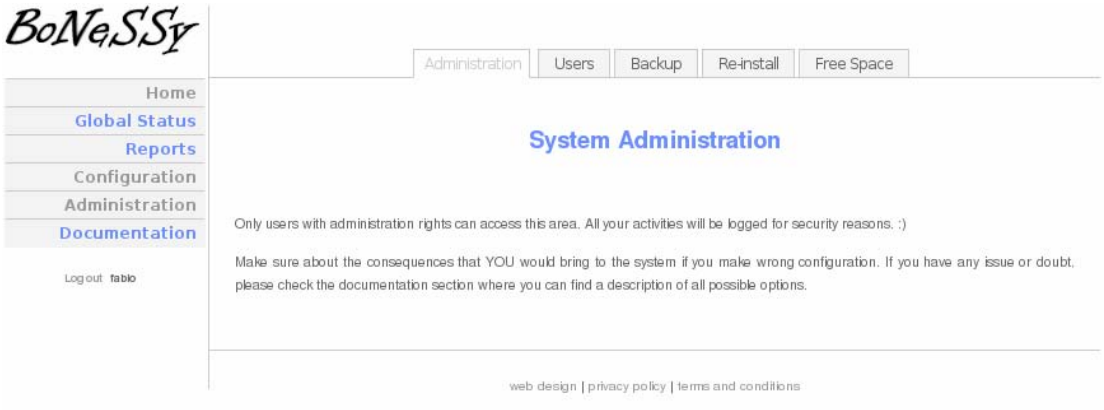

Figure 32 BoNeSSy main administration page

At the administration area, it is possible to add users, backup the whole system, re-install it and free some extra space.

It is very simple to add new users to the system. Just by clicking at the “Users” sub-menu at the top of the page, a new page with all the functions related to accounts administration will appear (figure 33). Here, the administrator can add or remove users and lock or unlock users. The lock feature can be used in case the administrator, for some reason, wants to disable the user account for some period of time.

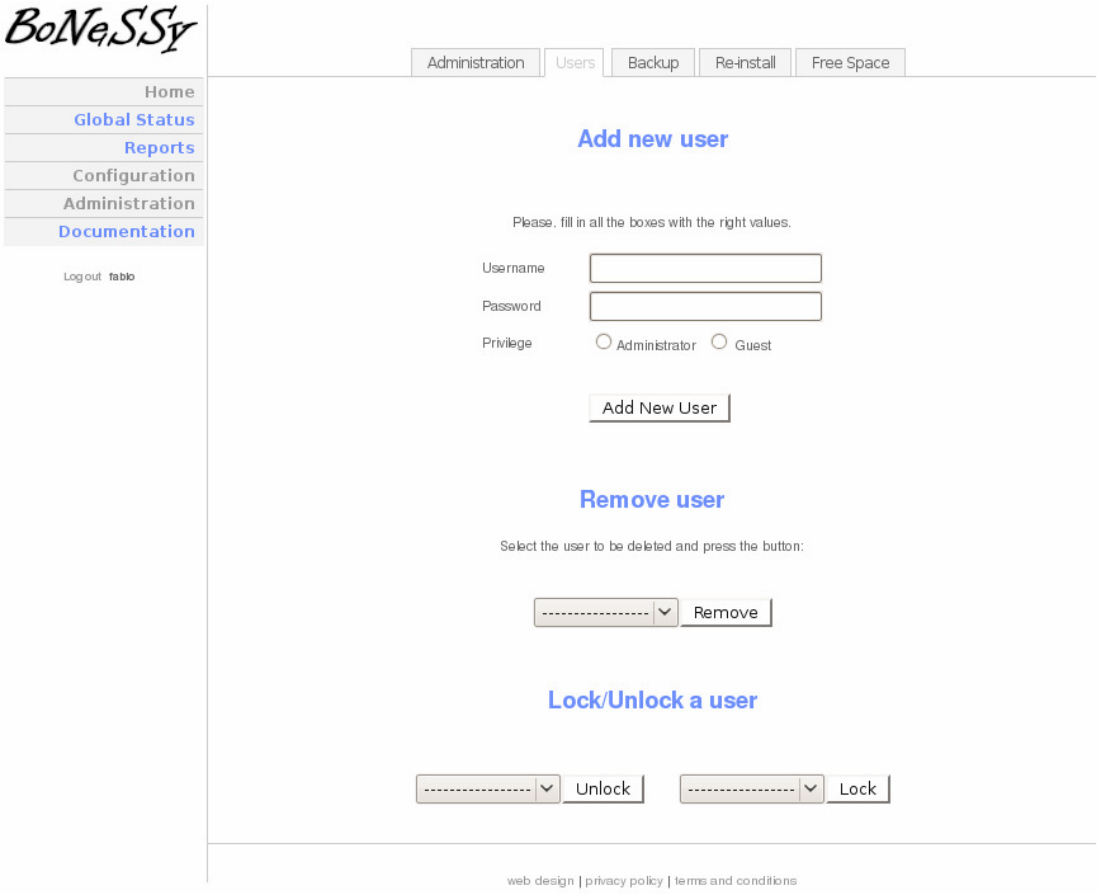

Figure 33 BoNeSSy add/remove/lock/unlock users page

The next functionality accessible through the administration menu is the backup/restore feature. There, it is possible with a single click to backup all system (figure 34). By this whole system backup we mean:

- The whole database structure and data;
- The system configuration files.

The output of the system backup will be a file named *Bonessy_backup-YYYYMMDDhhmmss.tar.gz* that will be located at the *backup* folder inside the BoNeSSy home directory.

The restore option is also available. User should browser the backup file he wants to restore and press the “Restore System” button (figure 34).

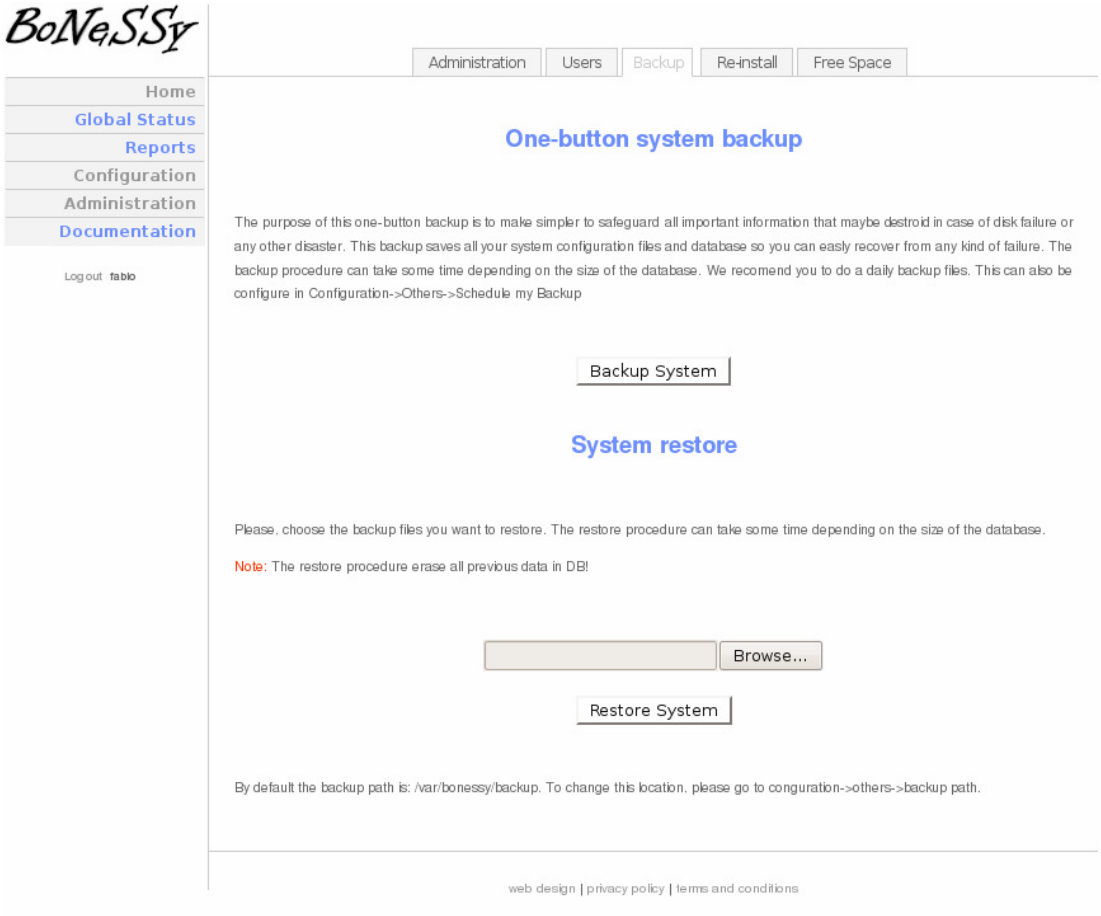

Figure 34 BoNeSSy backup page

For testing purposes, or when the system gets not functional, it is possible to reinstall the system from the root (figure 35). This means that all the database and system configuration files will be reset to default. It is apparent that this button is very dangerous and needs to be used very carefully.

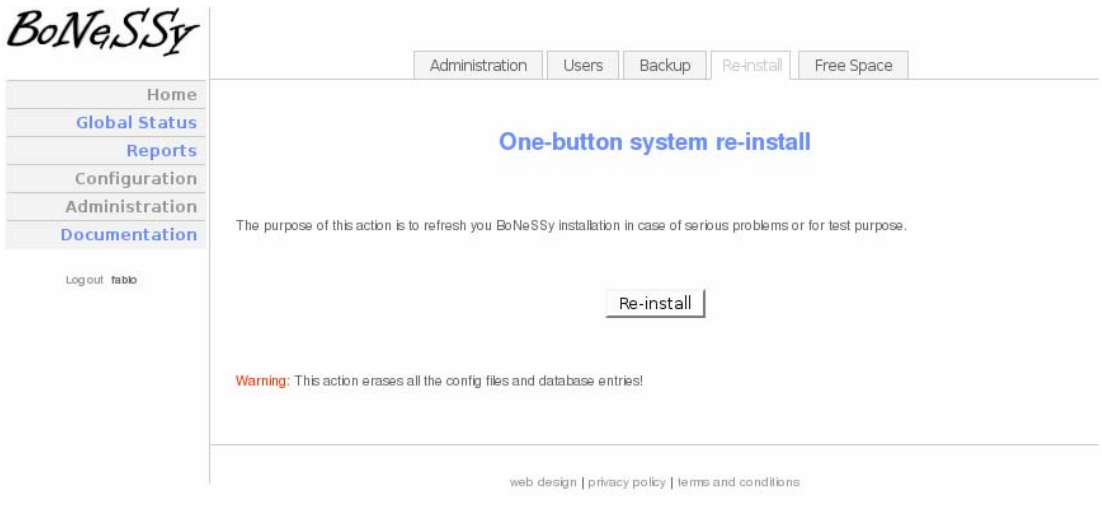

Figure 35 BoNeSSy re-install page

The last administration feature is the “Free Space” option (figure 36). There, the user can free some storage space by deleting all logs and tested files.

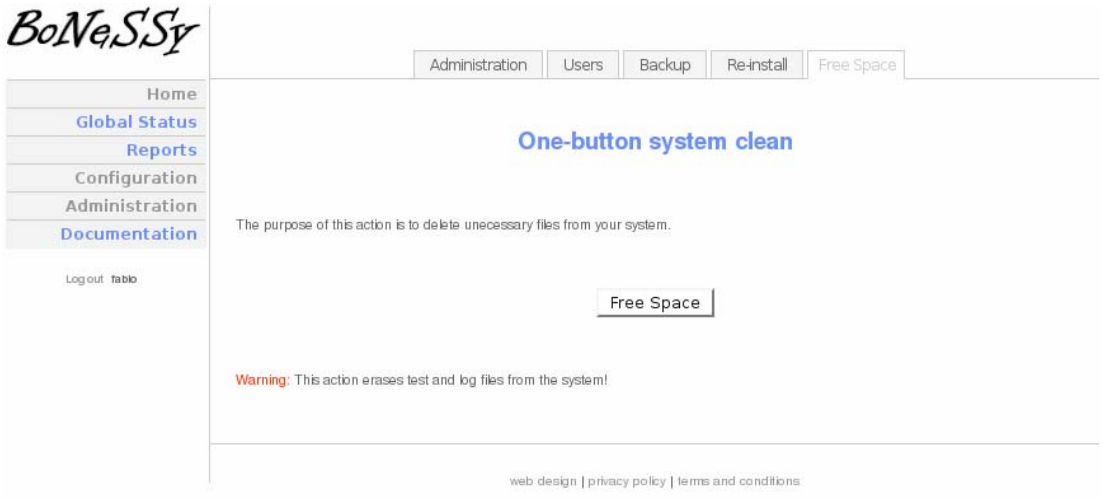

Figure 36 BoNeSSy cleanup page

The last button option at the left menu will lead the user to the documentation page (figure 37), where a system administration guide, the system architecture and the database scheme can be consulted.

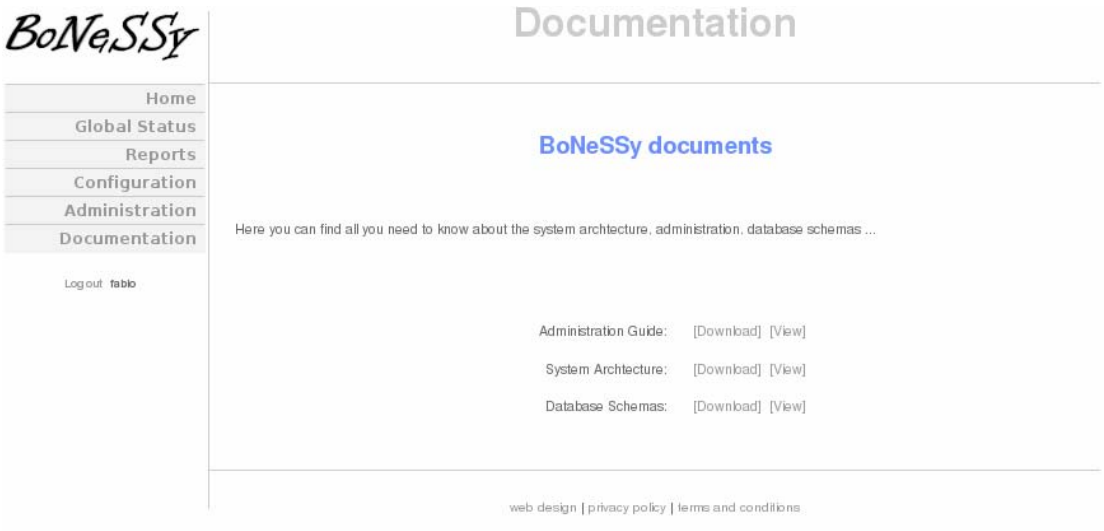

Figure 37 BoNeSSy documentation page

As can be seen in all other snapshots, the “log out” link is always available. Through this button the user will be able to safely log out the GUI by closing the PHP session. After this, the page will be redirected to the login page (figure 13).

3.6 Integration

In this sub-section we will discuss the scripts that enrich and integrate the system functionalities. There are several modules that need to be integrated, from the probes to the program interface. These scripts where defined at the system architecture as “System Global Processes” and “System Guard Monitor”.

It is important to understand the system folder structure before starting. There is a home folder: /opt/bonessy and inside it there are other folders for specific cases:

- Backup ($BONESSY_HOME/backup);
- Neural Network ($BONESSY_HOME/NN);
- Binary ($BONESSY_HOME/bin);
- Log ($BONESSY_HOME/log;
- Configuration ($BONESSY_HOME/config)

At /opt we have the bonessy.env file that configures all the system global paths and environment variables, so the scripts can use these variables instead of hardcoded paths.

The input files that arrive from the probes will be stored at /data/inputs and after serving as input to the neural network, they will be compressed and stored at /data/backup with their original name but adding the .tar.gz suffix without their original

extension (ip-xxx_xxx_xxx_xxx_YYYYMMDDhhmmss.tar.gz). The output from the neural network will be placed at /data/nn_output and, after being inserted in the database, compressed and stored at the /data/nn_output_backup folder and renamed to the name (nn_output_ip-xxx_xxx_xxx_xxx_YYYYMMDDhhmmss.tar.gz).

Let's start from the probe and then go to the GUI. The program used for capturing the traffic (*trafana_lite*) doesn't have any *FTP* or *SSH* functionality to upload the captured files to the server. Thinking on this, we have a script named *bonessy_probe.sh* that is responsible for this task. It simply executes the *scp* command for copying the files to the appropriate folder at the main server and its work-flow is described at figure 38. Before the system is put into use, it is important to share the SSH keys, so that the script won't prompt for any password. The time interval between SCP's are 15 minutes and it is controlled by *crontab*.

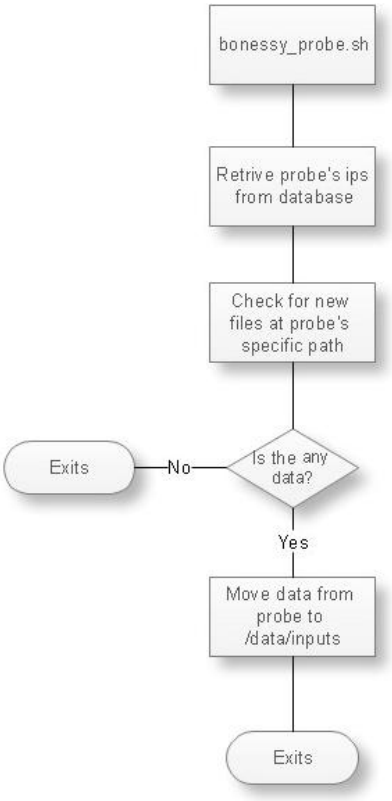

Figure 38 bonessy_probe.sh work-flow

For controlling the ANN (*bonessy_ann_test*), we have another script named bonessy_ann.sh. This script is always running in memory and it is responsible for running the ANN with the correct inputs. It monitors the input data folder for new files; after a new file arrives, it runs the ANN with the file and then copies a

compressed format of the file to a backup folder. Its work-flow can be seen at figure 39.

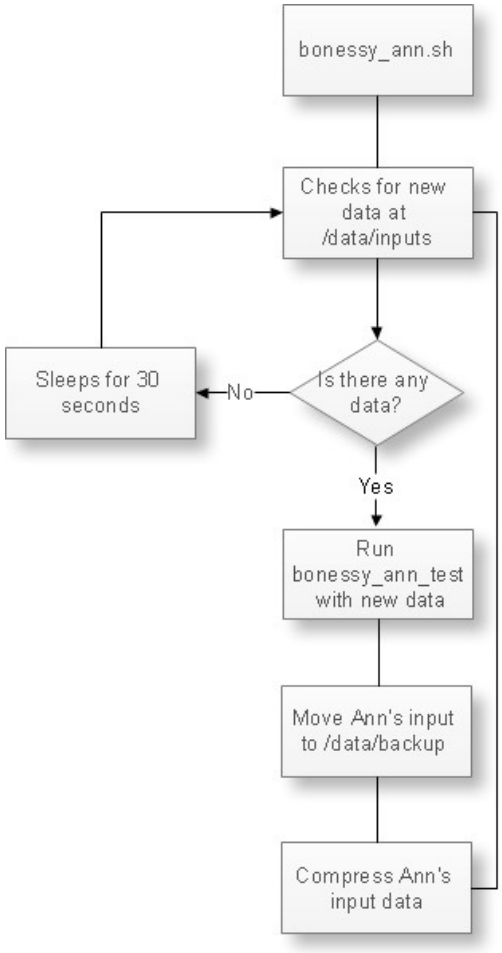

Figure 39 bonessy_ann.sh work-flow

The next script is the one responsible to parse the output of the artificial neural network and add it to the database (*bonessy_db_insert.sh*). Before inserting it at the database, the script calculates the percentage of anomalies, by counting the number of outputs present at the file divided by the number of ones that were found. After this, a simple insert on the tested table is done and the data used at this operation is compressed. See figure 40 for checking its work-flow.

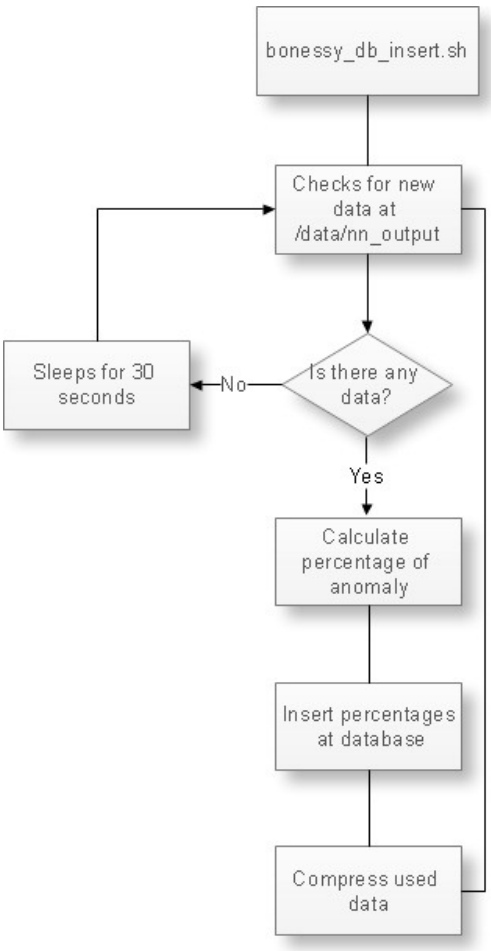

Figure 40 bonessy_db_insert.sh work-flow

To help the backup utility, we have a script named *bonessy_backup.sh* that exports the database, system configuration files and ANN configuration into the *backup* folder located at the *BoNeSSy home directory* (figure 41). This script uses the utility provided by MySQL to backup the database. The command is listed below:

```
/usr/bin/mysqldump --opt -h $hostname_db -u $username_db -
p$password_db $database | gzip > $backupFile
```

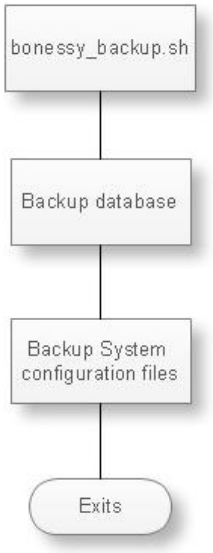

Figure 41 bonessy_backup.sh work-flow

Another functionality provided by the GUI is the system re-install. To reinstall a fresh system, we have a shell script named *bonessy_reinstall.sh* (figure 42). This script restores all system configuration and database. It is very useful for testing purposes.

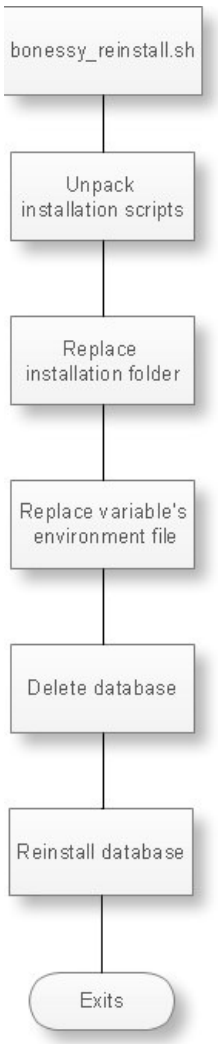

Figure 42 bonessy_reinstall.sh work-flow

For the system to have the blocking IP counter measurement and the deep trace functionalities, a script named *bonessy_counter.sh* was implemented (figure 43). For the IP blocking option, this script connects to the router through SSH and does the following command:

```
Access-list 101 deny ip xxx.xxx.xxx.xxx 255.255.255.255 any
```

The tracing option is used with the help of the *tshark* tool that is provided in the wireshark package and must be installed in the probe if this functionality is required. With this tool, it is possible to capture all activity from a specific IP Address so it can be seen in any graphical program that open files with the .cap extension.

The script *bonessy_counter.sh* is run with the following arguments:

```
./bonessy_counter.sh [type] [IP Address of client] [IP Adress of
probe or router]
```

where the type is 0 if the specific IP Address is to be put into deep analysis or 1 if the client IP Address should be blocked. For security reasons, there is no password stored from routers or probes, so the SSH keys must be shared.

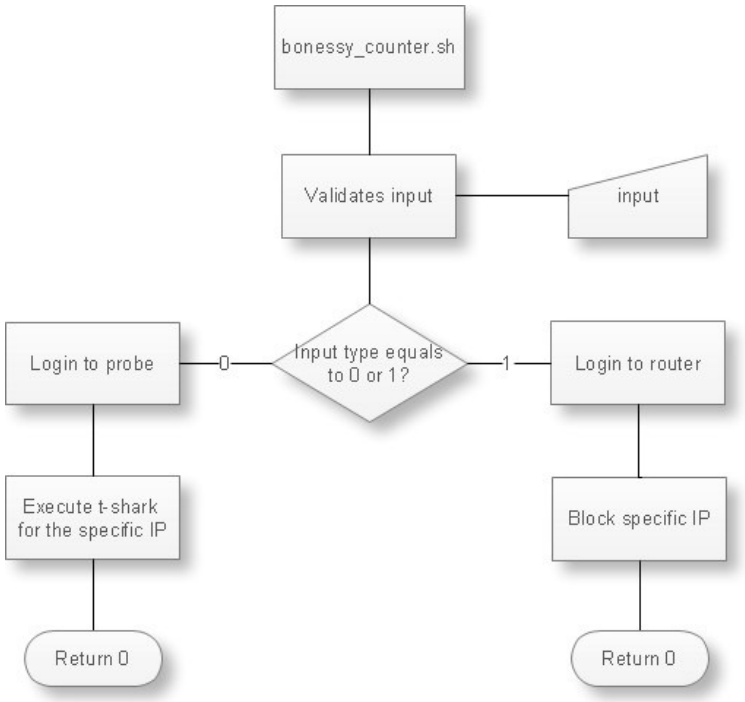

Figure 43 bonessy_counter.sh work-flow

Another important script is the one that is responsible to keep the system clean. This script is known as *bonessy_clean.sh* (figure 44). It is issued every midnight by *crontab*, reads the parameters set by the user from the *system_control* table and deletes any raw data file (that has the network traffic information) and ANN training file that is older than the specified time. It cleans the following tables from the database: *alarm_monitor*, *tested* and *train*.

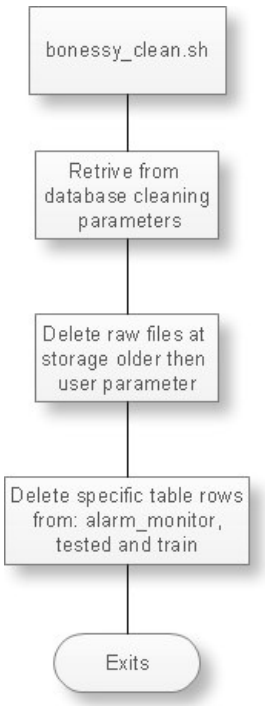

Figure 44 bonessy_clean.sh work-flow

To prevent any system vital process to shutdown we have the *service_guard.sh* tool. This tool is also a shell script that monitors any process configured in its configuration file (*service_guard.cf*). By using a configuration file, we give some flexibility to *service_guard*. It has the following syntax:

```
[program_name] [startup_command]
```

It works by monitoring system processes and logging its activity into *$BONESSY_HOME/logs/service_guard.log.* Its work-flow can be seen at figure 45.

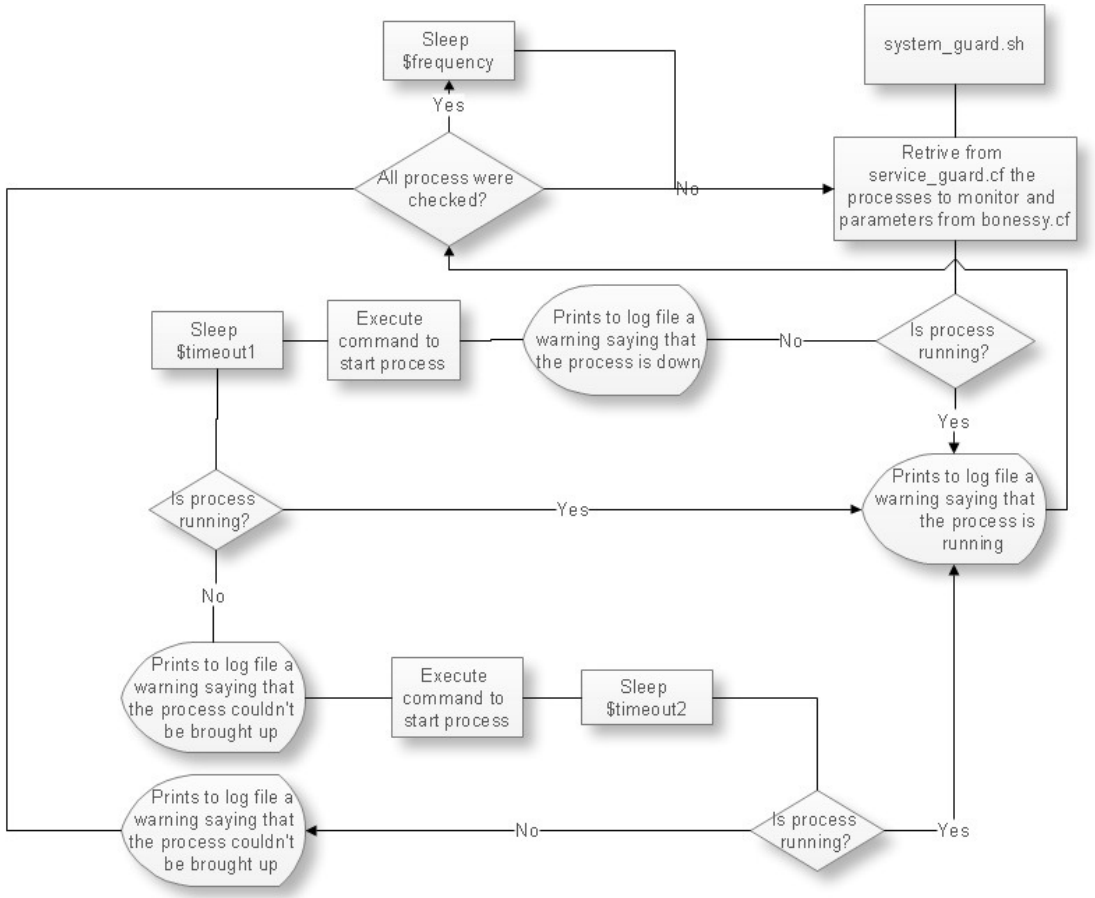

Figure 45 service_guard.sh work-flow

3.7 Summary

This chapter explained the whole system development. The development of the ANN was the most time consuming topic, since it took us time to find a reliable library, to understand it and to put it into use. The GUI development was also laborious because of the plethora of web pages and functionalities that were added.

4 SYSTEM TESTING AND VALIDATION

During this chapter, some parts of the system will be validated and tested in order to prove its effectiveness. As it is a big system, only the main and most important parts will be tested. The main modules are: capture probe, neural network, some integration parts and GUI.

4.1 Neural Network

One of the most common network protocols is HTTP. This protocol, as happens with others, has a characteristic traffic pattern that can be seen at figure 46.

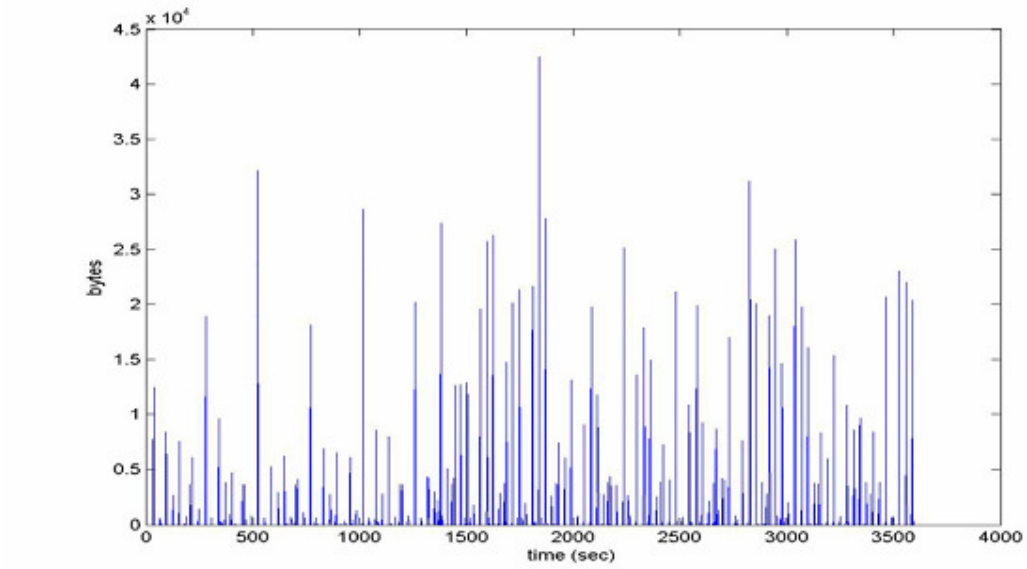

Figure 46 HTTP download traffic

This traffic is characterized by peaks that are alternated with moments of silence.

For testing and validation of the core part of the system, we will use data that was captured in the University of Aveiro wireless network using a network device that has 54 Mbps of downstream/upstream capability. For the following experiments, a period of 3600 seconds was chosen with a 1 second sample time. The program responsible for capturing the network traffic was *Trafana Lite*, a previous program developed in C that uses *libpcap*. Each trace file has 1 hour of duration that represents the number of uploaded and downloaded bytes per sampling interval. In order to generate licit traffic, we used the *Firefox* browser with regular tasks, like checking emails and reading news. To generate illicit traffic, we used the *Subseven* Trojan. This tool was chosen because it was widely used, it is easy to find and has good functionalities. There are three main services we will

use from *Subseven*: port scan, snapshots and file transfer. These functionalities were chosen because the attacker firstly needs to identify his victims: that's why we used the port scan functionality. After identifying his victim, normally the attacker will steal some information by downloading some files or seeing the victim screen. We have simulated some possible user behaviors. Here follows the detailed explanation on how these illicit traffics scenarios were generated:

- Port scan: A port scan to the compromised PC was done using ports between 10 and 65535. From the beginning of the experiment until instant 900 seconds, a port scan was made with a 1 second periodicity; from 900 to 1800 seconds, with 2 seconds periodicity; from 1800 to 2700 seconds with 3 seconds periodicity and from 2700 to 3600 seconds with 5 seconds periodicity.
- File transfer: The victim's computer will be used as an FTP server. A file download and upload will be used for this simulation. From 0-1800 seconds a file will be uploaded to the compromised computer and from 1800-3600 seconds a file will be downloaded from the compromised computer.
- Snapshot (periodic): In this simulation, the compromised computer sends periodic snapshots. This periodic simulation was chosen because sometimes the attacker wants to use less network bandwidth to be less suspicious. So, from 0 to 900 seconds we used a 1 second periodicity, from 900 to 1800 seconds we used a 2 seconds periodicity, from 1800 to 2700 seconds a 3 seconds periodicity and from 2700 to 3600 seconds a 4 seconds periodicity.
- Snapshot (non-periodic): This simulation uses no periodicity between the screen shots but the screen shots have variable qualities. From 0 to 600 seconds a high quality snapshot was used, from 600 to 1200 seconds the quality was reduced by 50%, from 1200 to 1800 seconds a lower quality was used, from 1800 to 2400 seconds we used the highest quality again (like in the first 600 seconds), from 2400 to 3000 seconds half quality snapshots were used and from 3000 to 3600s the worst quality snapshots were used again.

For security reasons, the Virtual Box software from SUN was used to simulate both the attacker and the Zombie computers, enabling the creation of virtual machines.

To prepare the training and testing sets, we split each capture file into two equal parts; the first half will be used for training and the second half for testing purposes. After being divided in two, each file will be aggregated using the previous developed program with aggregation values of 2, 5 and 10 (a copy with no aggregation will be used as well). By an aggregation equal to 2 we mean summing two rows into one, by an aggregation equal to 5

we mean summing five rows into one and by an aggregation equal to 10 we mean summing ten rows into one. After aggregation, our files were formatted according to the FANN specification for testing and training, with $h = 5$, $h = 10$, $h = 15$.

In this case, no data normalization was needed, due to the fact that we only used one network device. For a better understanding about how data was transformed, see figure 47.

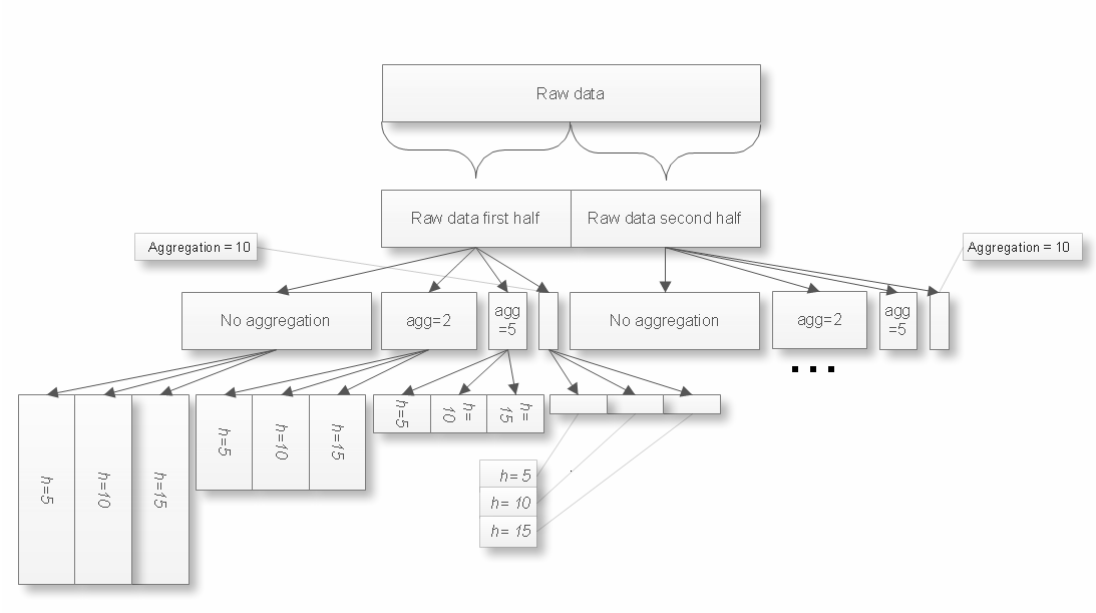

Figure 47 Transformation of the raw data into the training and testing sets

We divided the original data into many different formats to understand which one is more effective. We also used each data with different training algorithms (*Incremental, Batch, RPROP* and *QuickPROP*) to see which one is the most adequate to our case.

As this training will generate more than 200 results, we did a small script that will pick each file to train and test the ANN. This script will also make a graphic of each output file as well as calculate its percentage of success. These output files contain all neural network output values for each input file. Each input file has several vectors (*h*) inside that generate an output from 0 to 1 for each *h.* As the ANN algorithms work with supervised training, each licit input to the ANN has 0 as expected output value and each illicit input has 1 as expected output value. All the values bigger than 0.5 will be considered as illicit and all values smaller or equal to 0.5 will be considered as licit.

The best results were obtained using the training/testing sets with no aggregation and *h*=5, with aggregation equal to 0 or 2, using any sets of hidden neurons (*h*, 1.5**h*, 2**h*) and using the *QuickPROP* or *RPROP* training algorithm. As we can see from the results

illustrated at figures 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67 and table 1, the algorithm *QuickPROP* is the best in our case, giving a good balance between the false positives for licit and illicit traffic. The reason why the *RPROP* algorithm is listed on table 1 is because it was the one with less number of illicit false positives.

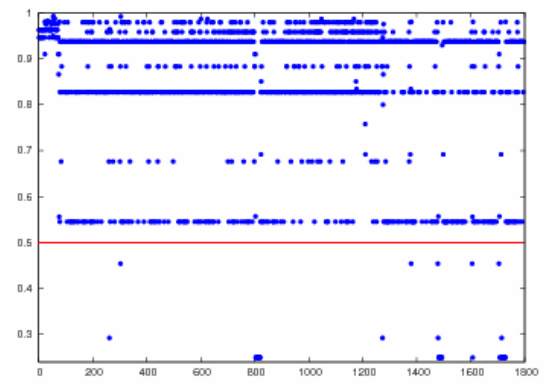

Figure 48 ANN output from HTTP traffic + file transfer with aggregation equal to 0, h = 5, hidden neurons = *h* and *QuickPROP* algorithm

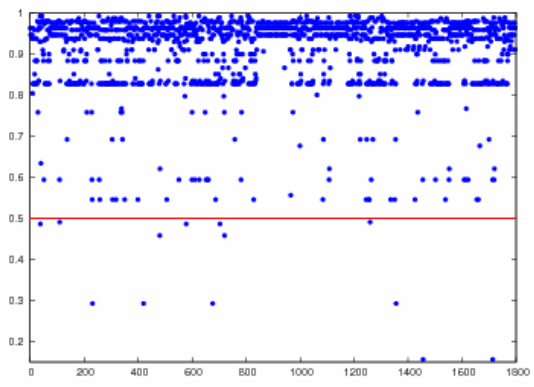

Figure 49 ANN output from HTTP traffic + port scan with aggregation equal to 0, *h* = 5, hidden neurons = *h* and QuickPROP algorithm

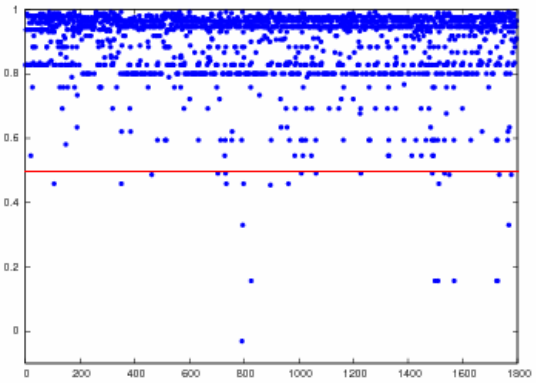

Figure 50 ANN output from HTTP traffic + periodical snapshot with aggregation equal to 0, *h* = 5, hidden neurons = *h* and QuickPROP algorithm

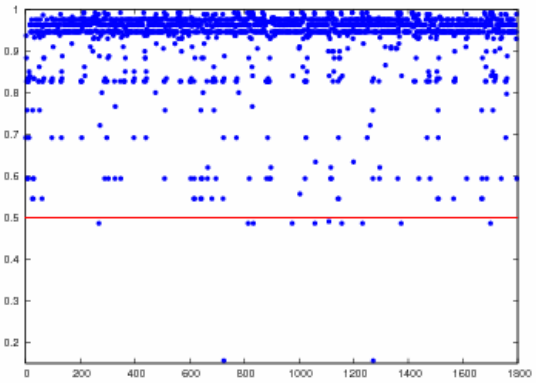

Figure 51 ANN output from HTTP traffic + non-periodical snapshot with aggregation equal to 0, *h* = 5, hidden neurons = *h* and QuickPROP algorithm

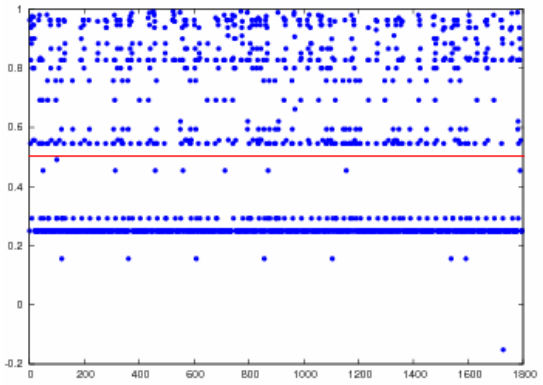

Figure 52 ANN output from HTTP licit traffic with aggregation equal to 0, *h* = 5, hidden neurons = *h* and QuickPROP algorithm

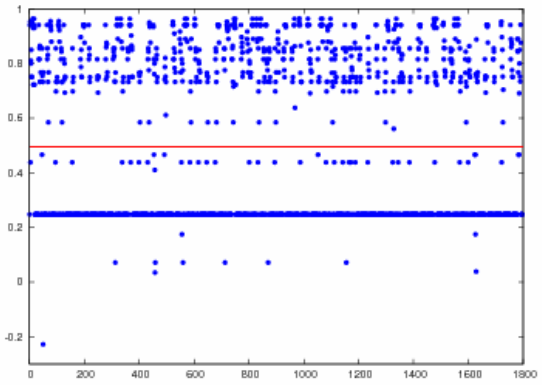

Figure 53 ANN output from HTTP licit traffic with aggregation equal to 0, *h* = 5, hidden neurons = 1,5**h* and QuickPROP algorithm

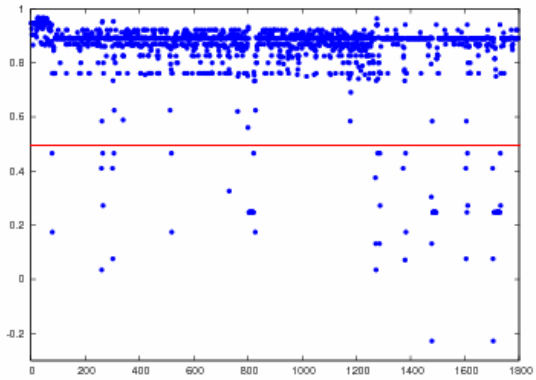

Figure 54 ANN output from HTTP traffic + file transfer with aggregation equal to 0, *h* = 5, hidden neurons = 1,5**h* and QuickPROP algorithm

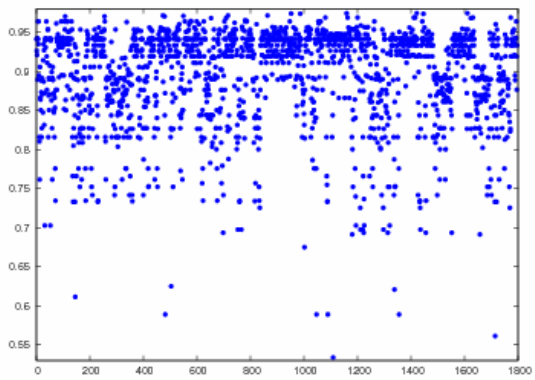

Figure 55 ANN output from HTTP traffic + port scan with aggregation equal to 0, *h* = 5, hidden neurons = 1,5**h* and QuickPROP algorithm

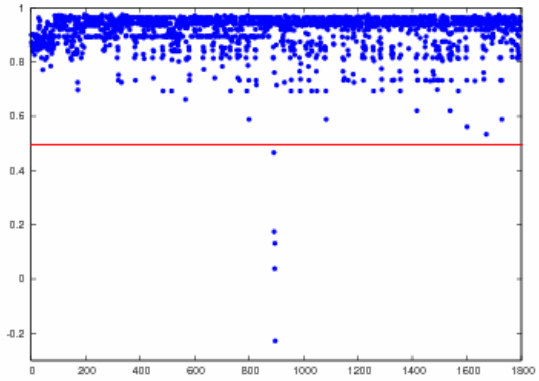

Figure 56 ANN output from *HTTP* traffic + periodic snapshoot with aggregation equal to 0, *h* = 5, hidden neurons = 1,5**h* and QuickPROP algorithm

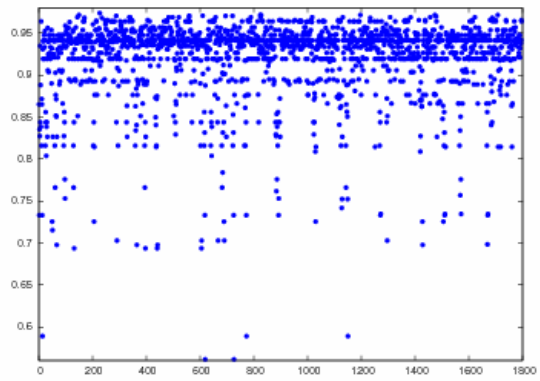

Figure 57 ANN output from HTTP traffic + non periodic snapshoot with aggregation equal to 0, *h* = 5, hidden neurons = 1,5**h* and QuickPROP algorithm

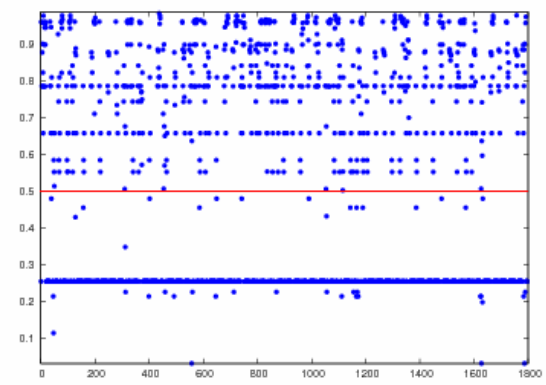

Figure 58 ANN output from *HTTP* licit traffic with aggregation equal to 0, $h = 5$, hidden neurons = $2*h$ and QuickPROP algorithm

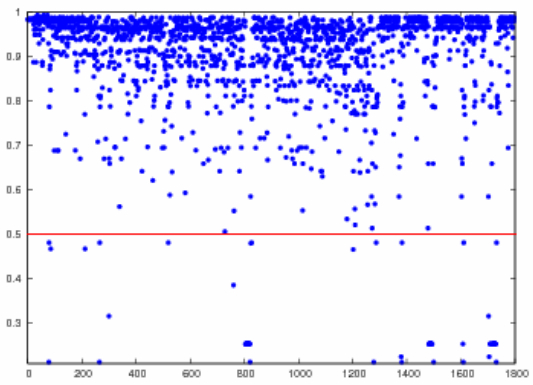

Figure 59 ANN output from HTTP traffic + file transfer with aggregation equal to 0, $h = 5$, hidden neurons = $2*h$ and QuickPROP algorithm

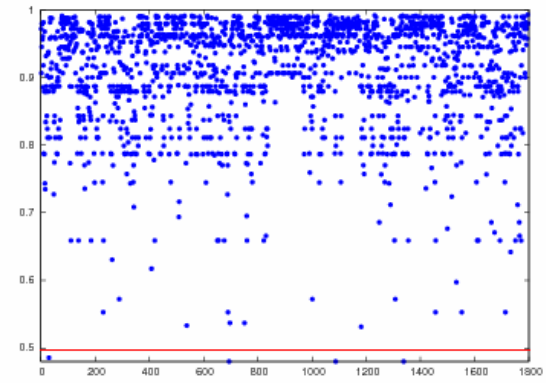

Figure 60 ANN output from *HTTP* traffic + port scan with aggregation equal to 0, $h = 5$, hidden neurons = $2*h$ and QuickPROP algorithm

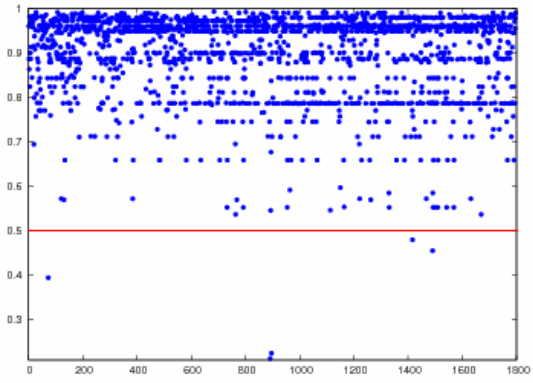

Figure 61 ANN output from HTTP traffic + periodic snapshoot with aggregation equal to 0, $h = 5$, hidden neurons = $2*h$ and QuickPROP algorithm

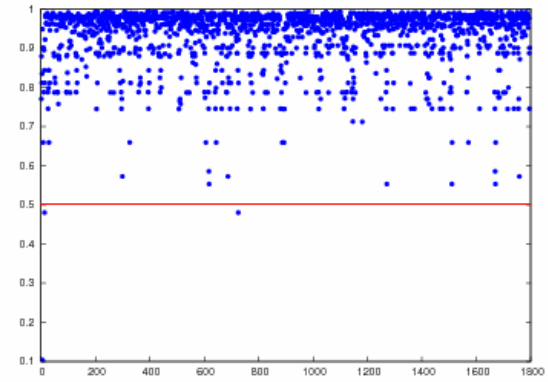

Figure 62 ANN output from *HTTP* traffic + non-periodic snapshoot with aggregation equal to 0, $h = 5$, hidden neurons = $2*h$ and QuickPROP algorithm

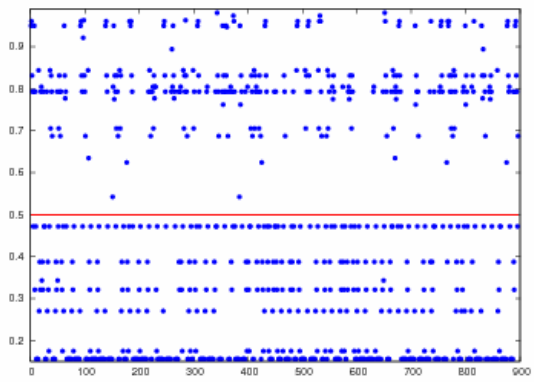

Figure 63 ANN output from HTTP licit traffic with aggregation equal to 2, $h = 5$, hidden neurons = $2*h$ and RPROP algorithm

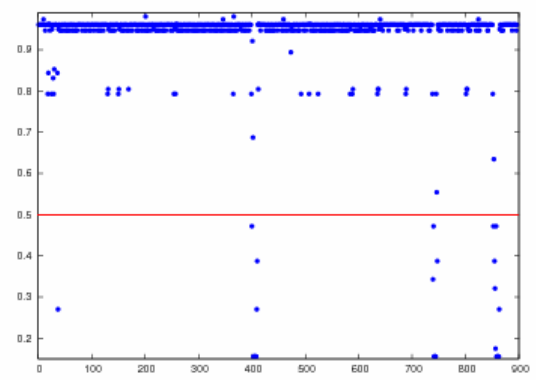

Figure 64 ANN output from *HTTP* traffic + file transfer with aggregation equal to 2, h = 5, hidden neurons = 2**h* and RPROP algorithm

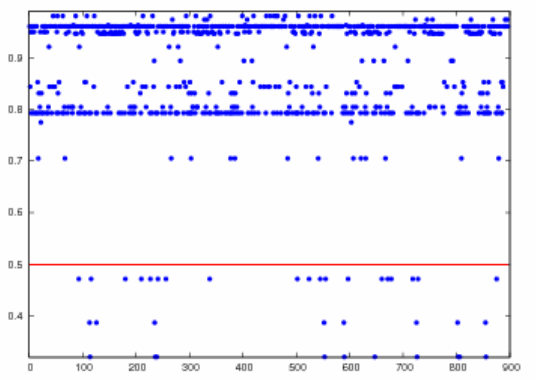

Figure 65 ANN output from HTTP traffic + port scan with aggregation equal to 2, h = 5, hidden neurons = 2**h* and RPROP algorithm

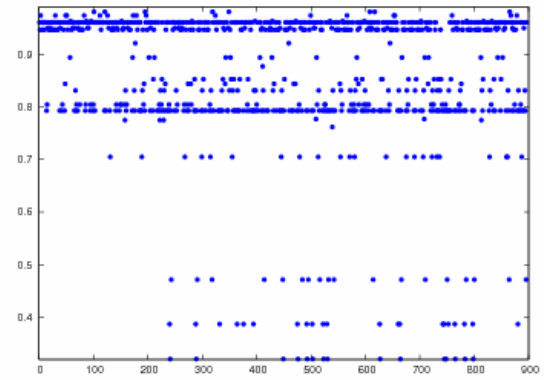

Figure 66 ANN output from *HTTP* traffic + periodic snapshoot with aggregation equal to 2, h = 5, hidden neurons = 2**h* and RPROP algorithm

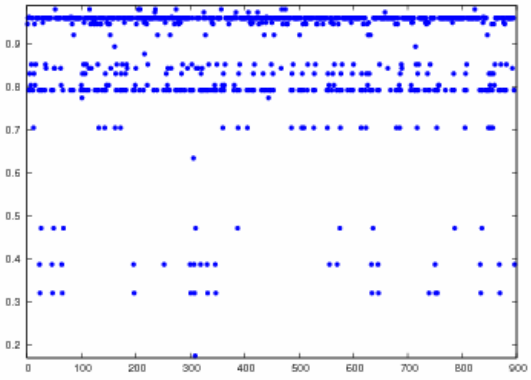

Figure 67 ANN output from HTTP traffic + non periodic snapshoot with aggregation equal to 2, h = 5, hidden neurons = 2**h* and RPROP algorithm

Traffic type	*Aggregation*	*h*	*Hidden Neurons*	*Trainning algorithm*	*Correct (%)*	*False positives (%)*
HTTP Licit	0	5	h	QuickPROP	66,87	33,13
HTTP+File transfer	0	5	h	QuickPROP	96,88	3,12
HTTP+Port scan	0	5	h	QuickPROP	99,28	0,72
HTTP+non periodic snap.	0	5	h	QuickPROP	98,50	1,50
HTTP+periodic snap.	0	5	h	QuickPROP	99,33	0,67
HTTP Licit	0	5	1,5*h	QuickPROP	64,20	35,80
HTTP+File transfer	0	5	1,5*h	QuickPROP	95,32	4,68
HTTP+Port scan	0	5	1,5*h	QuickPROP	100,00	0,00
HTTP+non periodic snap.	0	5	1,5*h	QuickPROP	99,72	0,28
HTTP+periodic snap.	0	5	1,5*h	QuickPROP	100,00	0,00
HTTP Licit	0	5	2*h	QuickPROP	63,70	36,3
HTTP+File transfer	0	5	2*h	QuickPROP	95,99	4,01
HTTP+Port scan	0	5	2*h	QuickPROP	99,78	0,22
HTTP+non periodic snap.	0	5	2*h	QuickPROP	99,72	0,28
HTTP+periodic snap.	0	5	2*h	QuickPROP	99,83	0,17
HTTP Licit	2	5	2*h	RPROP	68,75	31,25
HTTP+File transfer	2	5	2*h	RPROP	96,88	3,13
HTTP+Port scan	2	5	2*h	RPROP	95,87	4,13
HTTP+periodic snapshoot	2	5	2*h	RPROP	93,97	6,03
HTTP+periodic snapshoot	2	5	2*h	RPROP	95,20	4,80

Table 2 Neural network percentage of identification for the HTTP test

4.2 Overall

After analyzing the Artificial Neural Network behavior, a complete top to bottom system test will be done in this chapter. A test environment will be set to fully test the data flow of the system. With this test we will be able to check if data is being correctly captured, treated, inserted and shown in the alarm monitor page.

Before starting to setup the network, the BoNeSSy neural network was set up with one of the parameters set that gave us good results in the previous section: *h*=5, aggregation=0, hidden neurons=*h* and training algorithm= QuickPROP. This NN was trained to identify Skype's licit traffic as well as licit traffic + port scan, licit traffic + file transfer and licit traffic + periodic snapshot.

To set up the illicit traffic, a machine having the *Subseven* server installed will be used, as in the previous chapter. This compromised machine, named *infected PC* at figure 71, will be using *Skype* at the same time as the *Hacker's PC* will be doing a port scan in a first moment, then a file transfer and finally a periodic snapshot. The definitions about the port scan and the file transfer will be the same as in the previous chapter, except regarding the traffic generation time that will be set to 600 seconds in contrast with the 3600 seconds that were used in the previous chapter.

A probe was also installed in the same network to capture traffic. It was set to capture only the traffic from the compromised PC and the sample time was set to be equal to 1 second, as in the previous chapter. This means that for 600 seconds of generated traffic from each type, one capture file will be created for each type of traffic. The BoNeSSy server is also located in the same network. The network setup can be seen at figure 68.

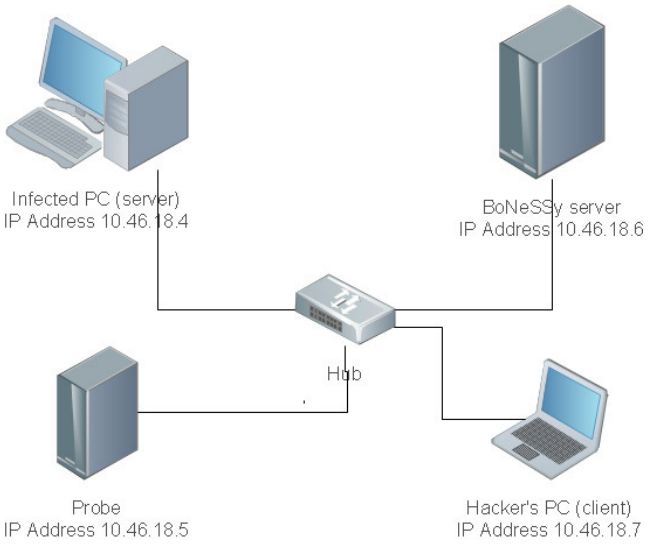

Figure 68 System test network

When the capture time was elapsed, we verified if the script responsible for transferring the captured data was working well. So, we checked the /data/nn_output_backup folder for files from the probe that had already been analyzed by the ANN and inserted at the database (figure 69).

```
nn_output_ip-192_168_2_100_20091129150001.tar.gz
nn_output_ip-192_168_2_100_20091129151001.tar.gz
nn_output_ip-192_168_2_100_20091129152001.tar.gz
nn_output_ip-192_168_2_100_20091129153001.tar.gz
```

Figure 69 Output from ls command at the /data/nn_output_backup folder

To build a table that can help us better understand the results, as in the previous chapter, we calculated the percentage of 0's and 1's by checking the files that were outputted by the ANN (table 2).

Traffic type	*Aggregation*	*h*	*Hidden Neurons*	*Trainning algorithm*	*Correct (%)*	*False positives (%)*
Skype Licit	0	5	*h*	QuickPROP	36,43	63,57
Skype+File transfer	0	5	*h*	QuickPROP	100,00	0,00
Skype+Port scan	0	5	*h*	QuickPROP	100,00	0,00
HTTP+periodic snap.	0	5	*h*	QuickPROP	100,00	0,00

Table 2 Neural network percentage of identification for the *Skype* test

The last step consists in verifying if the ANN output data is being correctly inserted in the database. As we have 4 traffic types, each with 600 seconds of duration, we should have 4 entries in the *tested* (figure 70) tables that correspond to each captured file from the probe.

We also set the anomaly alarm for the 10.46.18.0/255.255.255.0 network equal to 70% (figure 71), which means that only outputs from the ANN with a percentage that is higher than 70% will appear at the alarm monitor (figure 72).

```
mysql> select * from tested;
+---------+--------------------+------------+------------------+------------+
| test_id | networks_network_id | test_ip    | test_anom_percent | test_date  |
+---------+--------------------+------------+------------------+------------+
|       1 |                  3 | 10.46.18.4 |            63.57 | 2009-11-29 |
|       2 |                  3 | 10.46.18.4 |           100.00 | 2009-11-29 |
|       3 |                  3 | 10.46.18.4 |           100.00 | 2009-11-29 |
|       4 |                  3 | 10.46.18.4 |           100.00 | 2009-11-29 |
+---------+--------------------+------------+------------------+------------+
4 rows in set (0.00 sec)
```

Figure 70 Content of the tested table

```
mysql> select * from alarm_config;
+----------+--------------------+---------------+--------------------+---------------------+
| alarm_id | networks_network_id | users_user_id | alarm_percent_anom | alarm_data_add      |
+----------+--------------------+---------------+--------------------+---------------------+
|        1 |                  3 |             1 |              70.00 | 2009-11-29 00:00:00 |
+----------+--------------------+---------------+--------------------+---------------------+
1 row in set (0.00 sec)
```

Figure 71 Content of the alarm_config table

```
mysql> select * from alarm_monitor;
+------------+----------------------+---------------+--------------+----------------+-------------------------+
| monitor_id | alarm_config_alarm_id | tested_test_id | users_user_id | monitor_status | monitor_deactivate_date |
+------------+----------------------+---------------+--------------+----------------+-------------------------+
|          1 |                    1 |             2 |            1 |              1 | 2009-11-29              |
|          2 |                    1 |             3 |            1 |              1 | 2009-11-29              |
|          3 |                    1 |             4 |            1 |              1 | 2009-11-29              |
+------------+----------------------+---------------+--------------+----------------+-------------------------+
3rows in set (0.00 sec)
```

Figure 72 Content of the alarm_monitor table

To finalize our test, we checked the GUI to see if the data is being correctly displayed. Firstly, by checking the *System general healthy* page (figure 73), the *System alarms* (figure 74) page and the *Network alarm details* (figure 75) page.

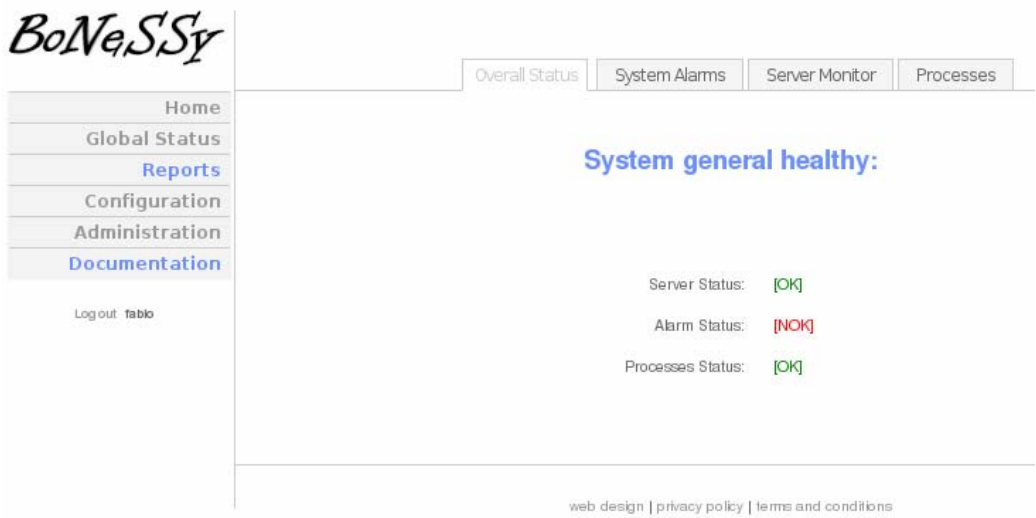

Figure 73 System general healthy page

Figure 74 System alarms page

Figure 75 Network alarm details

As we can conclude, the probe, the data work-flow, the database and the GUI are working well. The false positive results for the licit traffic are high, which can induct the system user to a false diagnose. Because of that, its alarm's thresholds should be high, near 80% of anomaly for a certain level of correctness.

5 CONCLUSION

This work presented a new methodology, based on neural networks, for detecting illicit traffic that traditional identification techniques are not able to identify since it is sensitive to traffic camouflage, like port change and cryptography. The results obtained show that we were able to achieve a high detection rate, of almost near 100%, of illicit traffic if the right parameters are set to the neural network model. This is a proof that this approach can be very effective where other methodologies fail. Furthermore, a joint effort between the new and the traditional techniques can bring system security to another level of efficiency. Nevertheless, great power usually brings great responsibility and the administrator of this IDS has to be careful when taking drastic measures against the computers that are responsible for the detected illicit traffic, since sometimes the system can give a high percentage of false positives.

Several authors seem to agree with the above results, explaining that neural networks can be a valuable tool to detect different licit network traffic, as well as illicit applications [24, 30, 36, 37].

As a commercial product, this methodology can have a bright future, since it does not rely on signature databases or specific patterns. It is self learning and new treats can be identified without any human intervention, besides the fact that it is not computationally heavy and can run into quite cheap servers. Nevertheless, the big percentage of false positives tells us that the algorithms used in this work are not the most suitable for this kind of pattern identification. This means that new algorithms need to be tested to reduce the high percentage of false positives, in order for the network administrator to fully trust on this methodology. The percentage of false positives are high probably because the ANN training algorithms are not the most suitable for detecting network traffic patterns or due to the fact that some parameters were not optimized, although we tested with different values of epochs, hidden neurons, hidden layers, *h* and aggregation. We didn't test with more ANN training algorithms because the library only supports the four types that were used.

6 FUTURE WORK

As future work, and in order to make the BoNeSSy system commercial, it is important to minimize the number of false positives that are given by the ANN while detecting licit traffic. In this way, other training algorithms (for example *Levenberg-Marquardt)* and/or ANN libraries must be tried. Regarding the system's interface, it is also important to add new functionalities, such as unlocking IP addresses, email alarms notifications and creating a RPM for easier system installation. Another important issue is to tune the database: since we didn't test the system in an exhaustive way, we didn't detect any slow performance caused by the database, but searching for DB bottlenecks and optimizing its performance using indexes or changing DB parameters is mandatory for achieving a high performance database and system.

References

1. OECD, *Malicious Software (Malware): A Security Threat to the internet economy*. 2007, Organisation for Economic Co-Operation and Development: Seoul, Korea.

2. Anderson, J.A., *An Introduction to Neural Networks* 1995: MIT Press. 666.

3. Jha, G.K., *Artificial Neural Networks and its applications*. 2007, IARI: New Delhi.

4. Gallant, S.I., *Neural Network Learning and Expert Systems*. 3 ed. 1995: MIT Press.

5. Nissen, S., *Implementation of a Fast Artificial Neural Network Library (fann)*. 2003, Department of Computer Science University of Copenhagen.

6. Priddy, K.L. and P.E. Keller, *Artificial neural networks: an introduction*. 2005: SPIE.

7. DTREG, *Multilayer Perceptron Neural Networks*. 2009.

8. Freeman, J.A. and D.M. Skapura, *Neural Networks Algorithms, Applications, and Programming Techniques*. 1991: Addison-Wesley Publishing Company.

9. Riedmiller, M. and H. Braun. *A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm*. in *IEEE International Conference On Neural Networks*. 1993. San Francisco, California.

10. Anastasiadis, A., G.D. Magoulas, and M.N. Vrahatis. *An Efficient Improvement of the Rprop Algorithm*. in *International Workshop on Artificial Neural Networks in Pattern Recognition*. 2003. Florence, Italy.

11. Micro, T., *Web Threat Watch USA*. 2007, Trend Micro.

12. Nickolov, E. *Modern trends in the cyber attacks against the critical information infrastructure*. in *Regional Cybersecurity Forum*. 2008. Sofia.

13. Stamp, M., *Information Security Principles and Practice*. 2006: John Wiley & Sons Inc.

14. Plant, R. and S. Murrell, *An Executive's Guide to Information Technology: Principles, Business Models, and Terminology*. 2007: Cambrigde University Press.

15. Tipton, H. and M. Krause, *Information Security Management Handbook*. Vol. 5. 2004: Auerbach Publications. 2036.

16. Raymond, R., *Malware development life cycle*, in *Virus Bulletin Conference*. 2008: Ottawa, Ontario, Canada. .

17. MIT. *What is a botnet?* 2009 [cited; Available from: http://kb.mit.edu/confluence/pages/viewpage.action?pageId=4270698.

18. Acohido, B., *Cybergangs use cheap labor to break codes on social sites*, in *USA Today*. 2009.

19. Bace, R., *Intrusion Detection*. 2000: Macmillan Technical Publishing.

20. Verwoerd, T. and R. Hunt, *Intrusion detection techniques and approaches.* Computer Communications, 2002. **25**(15): p. 1356-1365.

21. Debar, H., M. Becker, and D. Siboni, *A neural network component for an intrusion detection system.* Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on, 1992.

22. Lima, I.V.M.d., J.A. Degaspari, and J.B.M. Sobral. *Intrusion Detection Through Artificial Neural Networks*. in *Network Operations and Management Symposium*. 2008. Salvador, Brasil.

23. LI, J., G.-Y. ZHANG, and G.-C. GU. *THE RESEARCH AND IMPLEMENTATION OF INTELLIGENT INTRUSION DETECTION SYSTEM BASED ON ARTIFICIAL NEURAL NETWORK*. in *Third International Conference on Machine Laming and Cybernetics*. 2004. Shanghai, China.

24. Alhussein, M. and K. Al-Ghoneim, *Real-Time Network Intrusion Detection System Based*
on Neural Networks, in *4th International Conference On Enterprise Information Systems*. 2002: Ciudad Real - Spain.

25. Guan, D., et al. *A Collaborative Intrusion Detection System Using Log Server and Neural Networks*. in *International Conference on Mechatronics & Automation*. 2005. Niagara Falls, Canada.

26. Lippmann, R.P. and R.K. Cunningham, *Improving intrusion detection performance using keyword selection and neural networks*, in *RAID99*. 1999: Purdue, IN, USA

27. James, D.A., E.N. Richard, and M.H. Sallie, *A Procedural Approach to Evaluating Software Development Methodologies: The Foundation*. 1986, Virginia Polytechnic Institute \& State University.

28. Hessam, S.S., *On the Role of Quality Attributes in Specifying Software/System Architecture for Intelligent Systems Abstract*. 2008.

29. Gross, D. and E. Yu. *Evolving System Architecture to Meet Changing Business Goals: an Agent and*
Goal-Oriented Approach. in *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*. 2001. Toronto, Ont., Canada.

30. Salvador, P., et al. *Detection of illicit traffic using neural networks*. in *International Conference on Security and Cryptography*. 2008. Oporto, PORTUGAL: Insticc-Inst Syst Technologies Information Control & Communication.

31. Bishop, C.M., *Neural Networks for pattern recognition*. 1995 ed. 1995, NY, USA: Oxford University Press USA.

32. annpp, S. *ANN++ - Artificial Neural Networks in C++*. 2002 [cited; Available from: http://savannah.nongnu.org/projects/annpp/.

33. Nissen, S. *Fast Artificial Neural Network Library (FANN)*. 2009 [cited; Available from: http://leenissen.dk/fann/.

34. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning Representation by Backpropagation Errors*, ed. Nature. 1986.

35. Ballad, T. and W. Ballad, *Securing PHP Web Applications*. 2009.

36. Nogueira, A., P. Salvador, and R. Valadas, *Detecting Internet Applications using Neural Networks*, in *International conference Networking and Services*. 2006: Silicon Valley, California, USA.

37. Wang, J.-x., Z.-y. Wang, and K. Dai, *A network intrusion detection system based on the artificial neural networks*, in *Proceedings of the 3rd international conference on Information security*. 2004, ACM: Shanghai, China.

Appendix

Traffic type	Aggregation	*h*	Hidden Neurons	Trainning algorithm	x>=0.5	x<0.5	Total inputs	correct (%)	false positives (%)
HTTP Licit	0	10	h	Batch	1791	0	1791	0,00	100,00
HTTP+File transfer	0	10	h	Batch	1791	0	1791	100,00	0,00
HTTP+Port scan	0	10	h	Batch	1791	0	1791	100,00	0,00
HTTP+non periodic snap.	0	10	h	Batch	1791	0	1791	100,00	0,00
HTTP+periodic snap.	0	10	h	Batch	1791	0	1791	100,00	0,00
HTTP Licit	0	10	h	Incremental	1049	742	1791	41,43	58,57
HTTP+File transfer	0	10	h	Incremental	1762	29	1791	98,38	1,62
HTTP+Port scan	0	10	h	Incremental	1782	9	1791	99,50	0,50
HTTP+non periodic snap.	0	10	h	Incremental	1776	15	1791	99,16	0,84
HTTP+periodic snap.	0	10	h	Incremental	1783	8	1791	99,55	0,45
HTTP Licit	0	10	h	RPROP	799	992	1791	55,39	44,61
HTTP+File transfer	0	10	h	RPROP	1755	36	1791	97,99	2,01
HTTP+Port scan	0	10	h	RPROP	1755	36	1791	97,99	2,01
HTTP+non periodic snap.	0	10	h	RPROP	1722	69	1791	96,15	3,85
HTTP+periodic snap.	0	10	h	RPROP	1755	36	1791	97,99	2,01
HTTP Licit	0	10	h	QuickPROP	729	1062	1791	59,30	40,70
HTTP+File transfer	0	10	h	QuickPROP	1725	66	1791	96,31	3,69
HTTP+Port scan	0	10	h	QuickPROP	1755	36	1791	97,99	2,01
HTTP+non periodic snap.	0	10	h	QuickPROP	1729	62	1791	96,54	3,46
HTTP+periodic snap.	0	10	h	QuickPROP	1748	43	1791	97,60	2,40
HTTP Licit	0	15	h	Batch	1786	0	1786	0,00	100,00
HTTP+File transfer	0	15	h	Batch	1786	0	1786	100,00	0,00
HTTP+Port scan	0	15	h	Batch	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	h	Batch	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	h	Batch	1786	0	1786	100,00	0,00
HTTP Licit	0	15	h	Incremental	1351	435	1786	24,36	75,64
HTTP+File transfer	0	15	h	Incremental	1772	14	1786	99,22	0,78
HTTP+Port scan	0	15	h	Incremental	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	h	Incremental	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	h	Incremental	1786	0	1786	100,00	0,00
HTTP Licit	0	15	h	RPROP	1351	435	1786	24,36	75,64
HTTP+File transfer	0	15	h	RPROP	1772	14	1786	99,22	0,78
HTTP+Port scan	0	15	h	RPROP	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	h	RPROP	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	h	RPROP	1786	0	1786	100,00	0,00
HTTP Licit	0	15	h	QuickPROP	1351	435	1786	24,36	75,64
HTTP+File transfer	0	15	h	QuickPROP	1772	14	1786	99,22	0,78
HTTP+Port scan	0	15	h	QuickPROP	1786	0	1786	100,00	0,00

HTTP+non periodic snap.	0	15	h	QuickPROP	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	h	QuickPROP	1786	0	1786	100,00	0,00
HTTP Licit	0	5	h	Batch	1796	0	1796	0,00	100,00
HTTP+File transfer	0	5	h	Batch	1796	0	1796	100,00	0,00
HTTP+Port scan	0	5	h	Batch	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	h	Batch	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	h	Batch	1796	0	1796	100,00	0,00
HTTP Licit	0	5	h	Incremental	693	1103	1796	61,41	38,59
HTTP+File transfer	0	5	h	Incremental	1749	47	1796	97,38	2,62
HTTP+Port scan	0	5	h	Incremental	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	h	Incremental	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	h	Incremental	1796	0	1796	100,00	0,00
HTTP Licit	0	5	h	RPROP	687	1109	1796	61,75	38,25
HTTP+File transfer	0	5	h	RPROP	1739	57	1796	96,83	3,17
HTTP+Port scan	0	5	h	RPROP	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	h	RPROP	1793	3	1796	99,83	0,17
HTTP+periodic snap.	0	5	h	RPROP	1796	0	1796	100,00	0,00
HTTP Licit	0	5	h	QuickPROP	595	1201	1796	66,87	33,13
HTTP+File transfer	0	5	h	QuickPROP	1740	56	1796	96,88	3,12
HTTP+Port scan	0	5	h	QuickPROP	1783	13	1796	99,28	0,72
HTTP+non periodic snap.	0	5	h	QuickPROP	1769	27	1796	98,50	1,50
HTTP+periodic snap.	0	5	h	QuickPROP	1784	12	1796	99,33	0,67
HTTP Licit	10	15	h	Batch	166	0	166	0,00	100,00
HTTP+File transfer	10	15	h	Batch	166	0	166	100,00	0,00
HTTP+Port scan	10	15	h	Batch	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	h	Batch	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	h	Batch	166	0	166	100,00	0,00
HTTP Licit	10	15	h	Incremental	166	0	166	0,00	100,00
HTTP+File transfer	10	15	h	Incremental	166	0	166	100,00	0,00
HTTP+Port scan	10	15	h	Incremental	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	h	Incremental	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	h	Incremental	166	0	166	100,00	0,00
HTTP Licit	10	15	h	RPROP	166	0	166	0,00	100,00
HTTP+File transfer	10	15	h	RPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	h	RPROP	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	h	RPROP	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	h	RPROP	166	0	166	100,00	0,00
HTTP Licit	10	15	h	QuickPROP	166	0	166	0,00	100,00
HTTP+File transfer	10	15	h	QuickPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	h	QuickPROP	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	h	QuickPROP	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	h	QuickPROP	166	0	166	100,00	0,00
HTTP Licit	10	5	h	Batch	176	0	176	0,00	100,00

HTTP+File transfer	10	5	h	Batch	176	0	176	100,00	0,00
HTTP+Port scan	10	5	h	Batch	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	h	Batch	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	h	Batch	176	0	176	100,00	0,00
HTTP Licit	10	5	h	Incremental	176	0	176	0,00	100,00
HTTP+File transfer	10	5	h	Incremental	176	0	176	100,00	0,00
HTTP+Port scan	10	5	h	Incremental	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	h	Incremental	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	h	Incremental	176	0	176	100,00	0,00
HTTP Licit	10	5	h	RPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	h	RPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	h	RPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	h	RPROP	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	h	RPROP	176	0	176	100,00	0,00
HTTP Licit	10	5	h	QuickPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	h	QuickPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	h	QuickPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	h	QuickPROP	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	h	QuickPROP	176	0	176	100,00	0,00
HTTP Licit	2	10	h	Batch	891	0	891	0,00	100,00
HTTP+File transfer	2	10	h	Batch	891	0	891	100,00	0,00
HTTP+Port scan	2	10	h	Batch	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	h	Batch	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	h	Batch	891	0	891	100,00	0,00
HTTP Licit	2	10	h	Incremental	778	113	891	12,68	87,32
HTTP+File transfer	2	10	h	Incremental	890	1	891	99,89	0,11
HTTP+Port scan	2	10	h	Incremental	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	h	Incremental	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	h	Incremental	891	0	891	100,00	0,00
HTTP Licit	2	10	h	RPROP	778	113	891	12,68	87,32
HTTP+File transfer	2	10	h	RPROP	890	1	891	99,89	0,11
HTTP+Port scan	2	10	h	RPROP	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	h	RPROP	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	h	RPROP	891	0	891	100,00	0,00
HTTP Licit	2	10	h	QuickPROP	778	113	891	12,68	87,32
HTTP+File transfer	2	10	h	QuickPROP	890	1	891	99,89	0,11
HTTP+Port scan	2	10	h	QuickPROP	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	h	QuickPROP	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	h	QuickPROP	891	0	891	100,00	0,00
HTTP Licit	2	15	h	Batch	886	0	886	0,00	100,00
HTTP+File transfer	2	15	h	Batch	886	0	886	100,00	0,00
HTTP+Port scan	2	15	h	Batch	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	h	Batch	886	0	886	100,00	0,00

HTTP+periodic snap.	2	15	h	Batch	886	0	886	100,00	0,00
HTTP Licit	2	15	h	Incremental	886	0	886	0,00	100,00
HTTP+File transfer	2	15	h	Incremental	886	0	886	100,00	0,00
HTTP+Port scan	2	15	h	Incremental	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	h	Incremental	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	h	Incremental	886	0	886	100,00	0,00
HTTP Licit	2	15	h	RPROP	886	0	886	0,00	100,00
HTTP+File transfer	2	15	h	RPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	h	RPROP	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	h	RPROP	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	h	RPROP	886	0	886	100,00	0,00
HTTP Licit	2	15	h	QuickPROP	886	0	886	0,00	100,00
HTTP+File transfer	2	15	h	QuickPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	h	QuickPROP	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	h	QuickPROP	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	h	QuickPROP	886	0	886	100,00	0,00
HTTP Licit	2	5	h	Batch	896	0	896	0,00	100,00
HTTP+File transfer	2	5	h	Batch	896	0	896	100,00	0,00
HTTP+Port scan	2	5	h	Batch	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	h	Batch	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	h	Batch	896	0	896	100,00	0,00
HTTP Licit	2	5	h	Incremental	544	352	896	39,29	60,71
HTTP+File transfer	2	5	h	Incremental	881	15	896	98,33	1,67
HTTP+Port scan	2	5	h	Incremental	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	h	Incremental	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	h	Incremental	896	0	896	100,00	0,00
HTTP Licit	2	5	h	RPROP	492	404	896	45,09	54,91
HTTP+File transfer	2	5	h	RPROP	877	19	896	97,88	2,12
HTTP+Port scan	2	5	h	RPROP	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	h	RPROP	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	h	RPROP	896	0	896	100,00	0,00
HTTP Licit	2	5	h	QuickPROP	544	352	896	39,29	60,71
HTTP+File transfer	2	5	h	QuickPROP	881	15	896	98,33	1,67
HTTP+Port scan	2	5	h	QuickPROP	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	h	QuickPROP	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	h	QuickPROP	896	0	896	100,00	0,00
HTTP Licit	5	10	h	Batch	351	0	351	0,00	100,00
HTTP+File transfer	5	10	h	Batch	351	0	351	100,00	0,00
HTTP+Port scan	5	10	h	Batch	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	h	Batch	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	h	Batch	351	0	351	100,00	0,00
HTTP Licit	5	10	h	Incremental	351	0	351	0,00	100,00
HTTP+File transfer	5	10	h	Incremental	351	0	351	100,00	0,00

HTTP+Port scan	5	10	h	Incremental	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	h	Incremental	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	h	Incremental	351	0	351	100,00	0,00
HTTP Licit	5	10	h	RPROP	351	0	351	0,00	100,00
HTTP+File transfer	5	10	h	RPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	h	RPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	h	RPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	h	RPROP	351	0	351	100,00	0,00
HTTP Licit	5	10	h	QuickPROP	351	0	351	0,00	100,00
HTTP+File transfer	5	10	h	QuickPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	h	QuickPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	h	QuickPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	h	QuickPROP	351	0	351	100,00	0,00
HTTP Licit	5	15	h	Batch	346	0	346	0,00	100,00
HTTP+File transfer	5	15	h	Batch	346	0	346	100,00	0,00
HTTP+Port scan	5	15	h	Batch	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	h	Batch	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	h	Batch	346	0	346	100,00	0,00
HTTP Licit	5	15	h	Incremental	346	0	346	0,00	100,00
HTTP+File transfer	5	15	h	Incremental	346	0	346	100,00	0,00
HTTP+Port scan	5	15	h	Incremental	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	h	Incremental	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	h	Incremental	346	0	346	100,00	0,00
HTTP Licit	5	15	h	RPROP	346	0	346	0,00	100,00
HTTP+File transfer	5	15	h	RPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	h	RPROP	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	h	RPROP	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	h	RPROP	346	0	346	100,00	0,00
HTTP Licit	5	15	h	QuickPROP	346	0	346	0,00	100,00
HTTP+File transfer	5	15	h	QuickPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	h	QuickPROP	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	h	QuickPROP	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	h	QuickPROP	346	0	346	100,00	0,00
HTTP Licit	5	5	h	Batch	356	0	356	0,00	100,00
HTTP+File transfer	5	5	h	Batch	356	0	356	100,00	0,00
HTTP+Port scan	5	5	h	Batch	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	h	Batch	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	h	Batch	356	0	356	100,00	0,00
HTTP Licit	5	5	h	Incremental	339	17	356	4,78	95,22
HTTP+File transfer	5	5	h	Incremental	356	0	356	100,00	0,00
HTTP+Port scan	5	5	h	Incremental	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	h	Incremental	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	h	Incremental	356	0	356	100,00	0,00

HTTP Licit	5	5	h	RPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	h	RPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	h	RPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	h	RPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	h	RPROP	356	0	356	100,00	0,00
HTTP Licit	5	5	h	QuickPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	h	QuickPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	h	QuickPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	h	QuickPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	h	QuickPROP	356	0	356	100,00	0,00
HTTP Licit	0	10	1,5*h	Batch	1791	0	1791	0,00	100,00
HTTP+File transfer	0	10	1,5*h	Batch	1791	0	1791	100,00	0,00
HTTP+Port scan	0	10	1,5*h	Batch	1791	0	1791	100,00	0,00
HTTP+non periodic snap.	0	10	1,5*h	Batch	1791	0	1791	100,00	0,00
HTTP+periodic snap.	0	10	1,5*h	Batch	1791	0	1791	100,00	0,00
HTTP Licit	0	10	1,5*h	Incremental	993	798	1791	44,56	55,44
HTTP+File transfer	0	10	1,5*h	Incremental	1756	35	1791	98,05	1,95
HTTP+Port scan	0	10	1,5*h	Incremental	1780	11	1791	99,39	0,61
HTTP+non periodic snap.	0	10	1,5*h	Incremental	1782	9	1791	99,50	0,50
HTTP+periodic snap.	0	10	1,5*h	Incremental	1786	5	1791	99,72	0,28
HTTP Licit	0	10	1,5*h	RPROP	796	995	1791	55,56	44,44
HTTP+File transfer	0	10	1,5*h	RPROP	1754	37	1791	97,93	2,07
HTTP+Port scan	0	10	1,5*h	RPROP	1776	15	1791	99,16	0,84
HTTP+non periodic snap.	0	10	1,5*h	RPROP	1762	29	1791	98,38	1,62
HTTP+periodic snap.	0	10	1,5*h	RPROP	1769	22	1791	98,77	1,23
HTTP Licit	0	10	1,5*h	QuickPROP	853	938	1791	52,37	47,63
HTTP+File transfer	0	10	1,5*h	QuickPROP	1747	44	1791	97,54	2,46
HTTP+Port scan	0	10	1,5*h	QuickPROP	1759	32	1791	98,21	1,79
HTTP+non periodic snap.	0	10	1,5*h	QuickPROP	1744	47	1791	97,38	2,62
HTTP+periodic snap.	0	10	1,5*h	QuickPROP	1770	21	1791	98,83	1,17
HTTP Licit	0	15	1,5*h	Batch	1786	0	1786	0,00	100,00
HTTP+File transfer	0	15	1,5*h	Batch	1786	0	1786	100,00	0,00
HTTP+Port scan	0	15	1,5*h	Batch	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	1,5*h	Batch	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	1,5*h	Batch	1786	0	1786	100,00	0,00
HTTP Licit	0	15	1,5*h	Incremental	1351	435	1786	24,36	75,64
HTTP+File transfer	0	15	1,5*h	Incremental	1772	14	1786	99,22	0,78
HTTP+Port scan	0	15	1,5*h	Incremental	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	1,5*h	Incremental	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	1,5*h	Incremental	1786	0	1786	100,00	0,00
HTTP Licit	0	15	1,5*h	RPROP	760	1026	1786	57,45	42,55
HTTP+File transfer	0	15	1,5*h	RPROP	1757	29	1786	98,38	1,62
HTTP+Port scan	0	15	1,5*h	RPROP	1684	102	1786	94,29	5,71

HTTP+non periodic snap.	0	15	1,5*h	RPROP	1634	152	1786	91,49	8,51
HTTP+periodic snap.	0	15	1,5*h	RPROP	1666	120	1786	93,28	6,72
HTTP Licit	0	15	1,5*h	QuickPROP	895	891	1786	49,89	50,11
HTTP+File transfer	0	15	1,5*h	QuickPROP	1762	24	1786	98,66	1,34
HTTP+Port scan	0	15	1,5*h	QuickPROP	1727	59	1786	96,70	3,30
HTTP+non periodic snap.	0	15	1,5*h	QuickPROP	1668	118	1786	93,39	6,61
HTTP+periodic snap.	0	15	1,5*h	QuickPROP	1711	75	1786	95,80	4,20
HTTP Licit	0	5	1,5*h	Batch	1796	0	1796	0,00	100,00
HTTP+File transfer	0	5	1,5*h	Batch	1796	0	1796	100,00	0,00
HTTP+Port scan	0	5	1,5*h	Batch	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	1,5*h	Batch	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	1,5*h	Batch	1796	0	1796	100,00	0,00
HTTP Licit	0	5	1,5*h	Incremental	693	1103	1796	61,41	38,59
HTTP+File transfer	0	5	1,5*h	Incremental	1749	47	1796	97,38	2,62
HTTP+Port scan	0	5	1,5*h	Incremental	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	1,5*h	Incremental	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	1,5*h	Incremental	1796	0	1796	100,00	0,00
HTTP Licit	0	5	1,5*h	RPROP	690	1106	1796	61,58	38,42
HTTP+File transfer	0	5	1,5*h	RPROP	1748	48	1796	97,33	2,67
HTTP+Port scan	0	5	1,5*h	RPROP	1795	1	1796	99,94	0,06
HTTP+non periodic snap.	0	5	1,5*h	RPROP	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	1,5*h	RPROP	1796	0	1796	100,00	0,00
HTTP Licit	0	5	1,5*h	QuickPROP	643	1153	1796	64,20	35,80
HTTP+File transfer	0	5	1,5*h	QuickPROP	1712	84	1796	95,32	4,68
HTTP+Port scan	0	5	1,5*h	QuickPROP	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	1,5*h	QuickPROP	1791	5	1796	99,72	0,28
HTTP+periodic snap.	0	5	1,5*h	QuickPROP	1796	0	1796	100,00	0,00
HTTP Licit	10	10	1,5*h	Batch	171	0	171	0,00	100,00
HTTP+File transfer	10	10	1,5*h	Batch	171	0	171	100,00	0,00
HTTP+Port scan	10	10	1,5*h	Batch	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	1,5*h	Batch	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	1,5*h	Batch	171	0	171	100,00	0,00
HTTP Licit	10	10	1,5*h	Incremental	171	0	171	0,00	100,00
HTTP+File transfer	10	10	1,5*h	Incremental	171	0	171	100,00	0,00
HTTP+Port scan	10	10	1,5*h	Incremental	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	1,5*h	Incremental	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	1,5*h	Incremental	171	0	171	100,00	0,00
HTTP Licit	10	10	1,5*h	RPROP	171	0	171	0,00	100,00
HTTP+File transfer	10	10	1,5*h	RPROP	171	0	171	100,00	0,00
HTTP+Port scan	10	10	1,5*h	RPROP	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	1,5*h	RPROP	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	1,5*h	RPROP	171	0	171	100,00	0,00
HTTP Licit	10	10	1,5*h	QuickPROP	171	0	171	0,00	100,00

HTTP+File transfer	10	10	1,5*h	QuickPROP	171	0	171	100,00	0,00
HTTP+Port scan	10	10	1,5*h	QuickPROP	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	1,5*h	QuickPROP	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	1,5*h	QuickPROP	171	0	171	100,00	0,00
HTTP Licit	10	15	1,5*h	Batch	166	0	166	0,00	100,00
HTTP+File transfer	10	15	1,5*h	Batch	166	0	166	100,00	0,00
HTTP+Port scan	10	15	1,5*h	Batch	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	1,5*h	Batch	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	1,5*h	Batch	166	0	166	100,00	0,00
HTTP Licit	10	15	1,5*h	Incremental	166	0	166	0,00	100,00
HTTP+File transfer	10	15	1,5*h	Incremental	166	0	166	100,00	0,00
HTTP+Port scan	10	15	1,5*h	Incremental	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	1,5*h	Incremental	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	1,5*h	Incremental	166	0	166	100,00	0,00
HTTP Licit	10	15	1,5*h	RPROP	166	0	166	0,00	100,00
HTTP+File transfer	10	15	1,5*h	RPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	1,5*h	RPROP	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	1,5*h	RPROP	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	1,5*h	RPROP	166	0	166	100,00	0,00
HTTP Licit	10	15	1,5*h	QuickPROP	166	0	166	0,00	100,00
HTTP+File transfer	10	15	1,5*h	QuickPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	1,5*h	QuickPROP	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	1,5*h	QuickPROP	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	1,5*h	QuickPROP	166	0	166	100,00	0,00
HTTP Licit	10	5	1,5*h	Batch	176	0	176	0,00	100,00
HTTP+File transfer	10	5	1,5*h	Batch	176	0	176	100,00	0,00
HTTP+Port scan	10	5	1,5*h	Batch	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	1,5*h	Batch	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	1,5*h	Batch	176	0	176	100,00	0,00
HTTP Licit	10	5	1,5*h	Incremental	176	0	176	0,00	100,00
HTTP+File transfer	10	5	1,5*h	Incremental	176	0	176	100,00	0,00
HTTP+Port scan	10	5	1,5*h	Incremental	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	1,5*h	Incremental	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	1,5*h	Incremental	176	0	176	100,00	0,00
HTTP Licit	10	5	1,5*h	RPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	1,5*h	RPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	1,5*h	RPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	1,5*h	RPROP	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	1,5*h	RPROP	176	0	176	100,00	0,00
HTTP Licit	10	5	1,5*h	QuickPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	1,5*h	QuickPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	1,5*h	QuickPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	1,5*h	QuickPROP	176	0	176	100,00	0,00

HTTP+periodic snap.	10	5	1,5*h	QuickPROP	176	0	176	100,00	0,00
HTTP Licit	2	10	1,5*h	Batch	891	0	891	0,00	100,00
HTTP+File transfer	2	10	1,5*h	Batch	891	0	891	100,00	0,00
HTTP+Port scan	2	10	1,5*h	Batch	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	1,5*h	Batch	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	1,5*h	Batch	891	0	891	100,00	0,00
HTTP Licit	2	10	1,5*h	Incremental	778	113	891	12,68	87,32
HTTP+File transfer	2	10	1,5*h	Incremental	890	1	891	99,89	0,11
HTTP+Port scan	2	10	1,5*h	Incremental	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	1,5*h	Incremental	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	1,5*h	Incremental	891	0	891	100,00	0,00
HTTP Licit	2	10	1,5*h	RPROP	636	255	891	28,62	71,38
HTTP+File transfer	2	10	1,5*h	RPROP	888	3	891	99,66	0,34
HTTP+Port scan	2	10	1,5*h	RPROP	871	20	891	97,76	2,24
HTTP+non periodic snap.	2	10	1,5*h	RPROP	859	32	891	96,41	3,59
HTTP+periodic snap.	2	10	1,5*h	RPROP	863	28	891	96,86	3,14
HTTP Licit	2	10	1,5*h	QuickPROP	778	113	891	12,68	87,32
HTTP+File transfer	2	10	1,5*h	QuickPROP	890	1	891	99,89	0,11
HTTP+Port scan	2	10	1,5*h	QuickPROP	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	1,5*h	QuickPROP	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	1,5*h	QuickPROP	891	0	891	100,00	0,00
HTTP Licit	2	15	1,5*h	Batch	886	0	886	0,00	100,00
HTTP+File transfer	2	15	1,5*h	Batch	886	0	886	100,00	0,00
HTTP+Port scan	2	15	1,5*h	Batch	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	1,5*h	Batch	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	1,5*h	Batch	886	0	886	100,00	0,00
HTTP Licit	2	15	1,5*h	Incremental	886	0	886	0,00	100,00
HTTP+File transfer	2	15	1,5*h	Incremental	886	0	886	100,00	0,00
HTTP+Port scan	2	15	1,5*h	Incremental	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	1,5*h	Incremental	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	1,5*h	Incremental	886	0	886	100,00	0,00
HTTP Licit	2	15	1,5*h	RPROP	793	93	886	10,50	89,50
HTTP+File transfer	2	15	1,5*h	RPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	1,5*h	RPROP	879	7	886	99,21	0,79
HTTP+non periodic snap.	2	15	1,5*h	RPROP	879	7	886	99,21	0,79
HTTP+periodic snap.	2	15	1,5*h	RPROP	882	4	886	99,55	0,45
HTTP Licit	2	15	1,5*h	QuickPROP	886	0	886	0,00	100,00
HTTP+File transfer	2	15	1,5*h	QuickPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	1,5*h	QuickPROP	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	1,5*h	QuickPROP	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	1,5*h	QuickPROP	886	0	886	100,00	0,00
HTTP Licit	2	5	1,5*h	Batch	896	0	896	0,00	100,00
HTTP+File transfer	2	5	1,5*h	Batch	896	0	896	100,00	0,00

HTTP+Port scan	2	5	1,5*h	Batch	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	1,5*h	Batch	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	1,5*h	Batch	896	0	896	100,00	0,00
HTTP Licit	2	5	1,5*h	Incremental	462	434	896	48,44	51,56
HTTP+File transfer	2	5	1,5*h	Incremental	880	16	896	98,21	1,79
HTTP+Port scan	2	5	1,5*h	Incremental	879	17	896	98,10	1,90
HTTP+non periodic snap.	2	5	1,5*h	Incremental	871	25	896	97,21	2,79
HTTP+periodic snap.	2	5	1,5*h	Incremental	867	29	896	96,76	3,24
HTTP Licit	2	5	1,5*h	RPROP	481	415	896	46,32	53,68
HTTP+File transfer	2	5	1,5*h	RPROP	879	17	896	98,10	1,90
HTTP+Port scan	2	5	1,5*h	RPROP	882	14	896	98,44	1,56
HTTP+non periodic snap.	2	5	1,5*h	RPROP	874	22	896	97,54	2,46
HTTP+periodic snap.	2	5	1,5*h	RPROP	874	22	896	97,54	2,46
HTTP Licit	2	5	1,5*h	QuickPROP	493	403	896	44,98	55,02
HTTP+File transfer	2	5	1,5*h	QuickPROP	879	17	896	98,10	1,90
HTTP+Port scan	2	5	1,5*h	QuickPROP	890	6	896	99,33	0,67
HTTP+non periodic snap.	2	5	1,5*h	QuickPROP	887	9	896	99,00	1,00
HTTP+periodic snap.	2	5	1,5*h	QuickPROP	889	7	896	99,22	0,78
HTTP Licit	5	10	1,5*h	Batch	351	0	351	0,00	100,00
HTTP+File transfer	5	10	1,5*h	Batch	351	0	351	100,00	0,00
HTTP+Port scan	5	10	1,5*h	Batch	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	1,5*h	Batch	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	1,5*h	Batch	351	0	351	100,00	0,00
HTTP Licit	5	10	1,5*h	Incremental	351	0	351	0,00	100,00
HTTP+File transfer	5	10	1,5*h	Incremental	351	0	351	100,00	0,00
HTTP+Port scan	5	10	1,5*h	Incremental	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	1,5*h	Incremental	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	1,5*h	Incremental	351	0	351	100,00	0,00
HTTP Licit	5	10	1,5*h	RPROP	351	0	351	0,00	100,00
HTTP+File transfer	5	10	1,5*h	RPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	1,5*h	RPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	1,5*h	RPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	1,5*h	RPROP	351	0	351	100,00	0,00
HTTP Licit	5	10	1,5*h	QuickPROP	351	0	351	0,00	100,00
HTTP+File transfer	5	10	1,5*h	QuickPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	1,5*h	QuickPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	1,5*h	QuickPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	1,5*h	QuickPROP	351	0	351	100,00	0,00
HTTP Licit	5	15	1,5*h	Batch	346	0	346	0,00	100,00
HTTP+File transfer	5	15	1,5*h	Batch	346	0	346	100,00	0,00
HTTP+Port scan	5	15	1,5*h	Batch	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	1,5*h	Batch	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	1,5*h	Batch	346	0	346	100,00	0,00

HTTP Licit	5	15	1,5*h	Incremental	346	0	346	0,00	100,00
HTTP+File transfer	5	15	1,5*h	Incremental	346	0	346	100,00	0,00
HTTP+Port scan	5	15	1,5*h	Incremental	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	1,5*h	Incremental	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	1,5*h	Incremental	346	0	346	100,00	0,00
HTTP Licit	5	15	1,5*h	RPROP	346	0	346	0,00	100,00
HTTP+File transfer	5	15	1,5*h	RPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	1,5*h	RPROP	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	1,5*h	RPROP	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	1,5*h	RPROP	346	0	346	100,00	0,00
HTTP Licit	5	15	1,5*h	QuickPROP	346	0	346	0,00	100,00
HTTP+File transfer	5	15	1,5*h	QuickPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	1,5*h	QuickPROP	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	1,5*h	QuickPROP	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	1,5*h	QuickPROP	346	0	346	100,00	0,00
HTTP Licit	5	5	1,5*h	Batch	356	0	356	0,00	100,00
HTTP+File transfer	5	5	1,5*h	Batch	356	0	356	100,00	0,00
HTTP+Port scan	5	5	1,5*h	Batch	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	1,5*h	Batch	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	1,5*h	Batch	356	0	356	100,00	0,00
HTTP Licit	5	5	1,5*h	Incremental	339	17	356	4,78	95,22
HTTP+File transfer	5	5	1,5*h	Incremental	356	0	356	100,00	0,00
HTTP+Port scan	5	5	1,5*h	Incremental	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	1,5*h	Incremental	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	1,5*h	Incremental	356	0	356	100,00	0,00
HTTP Licit	5	5	1,5*h	RPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	1,5*h	RPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	1,5*h	RPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	1,5*h	RPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	1,5*h	RPROP	356	0	356	100,00	0,00
HTTP Licit	5	5	1,5*h	QuickPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	1,5*h	QuickPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	1,5*h	QuickPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	1,5*h	QuickPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	1,5*h	QuickPROP	356	0	356	100,00	0,00
HTTP Licit	0	10	2*h	Batch	1791	0	1791	0,00	100,00
HTTP+File transfer	0	10	2*h	Batch	1791	0	1791	100,00	0,00
HTTP+Port scan	0	10	2*h	Batch	1791	0	1791	100,00	0,00
HTTP+non periodic snap.	0	10	2*h	Batch	1791	0	1791	100,00	0,00
HTTP+periodic snap.	0	10	2*h	Batch	1791	0	1791	100,00	0,00
HTTP Licit	0	10	2*h	Incremental	820	971	1791	54,22	45,78
HTTP+File transfer	0	10	2*h	Incremental	1756	35	1791	98,05	1,95
HTTP+Port scan	0	10	2*h	Incremental	1757	34	1791	98,10	1,90

HTTP+non periodic snap.	0	10	2*h	Incremental	1734	57	1791	96,82	3,18
HTTP+periodic snap.	0	10	2*h	Incremental	1750	41	1791	97,71	2,29
HTTP Licit	0	10	2*h	RPROP	760	1031	1791	57,57	42,43
HTTP+File transfer	0	10	2*h	RPROP	1751	40	1791	97,77	2,23
HTTP+Port scan	0	10	2*h	RPROP	1761	30	1791	98,32	1,68
HTTP+non periodic snap.	0	10	2*h	RPROP	1738	53	1791	97,04	2,96
HTTP+periodic snap.	0	10	2*h	RPROP	1758	33	1791	98,16	1,84
HTTP Licit	0	10	2*h	QuickPROP	715	1076	1791	60,08	39,92
HTTP+File transfer	0	10	2*h	QuickPROP	1753	38	1791	97,88	2,12
HTTP+Port scan	0	10	2*h	QuickPROP	1743	48	1791	97,32	2,68
HTTP+non periodic snap.	0	10	2*h	QuickPROP	1701	90	1791	94,97	5,03
HTTP+periodic snap.	0	10	2*h	QuickPROP	1732	59	1791	96,71	3,29
HTTP Licit	0	15	2*h	Batch	1786	0	1786	0,00	100,00
HTTP+File transfer	0	15	2*h	Batch	1786	0	1786	100,00	0,00
HTTP+Port scan	0	15	2*h	Batch	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	2*h	Batch	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	2*h	Batch	1786	0	1786	100,00	0,00
HTTP Licit	0	15	2*h	Incremental	1351	435	1786	24,36	75,64
HTTP+File transfer	0	15	2*h	Incremental	1772	14	1786	99,22	0,78
HTTP+Port scan	0	15	2*h	Incremental	1786	0	1786	100,00	0,00
HTTP+non periodic snap.	0	15	2*h	Incremental	1786	0	1786	100,00	0,00
HTTP+periodic snap.	0	15	2*h	Incremental	1786	0	1786	100,00	0,00
HTTP Licit	0	15	2*h	RPROP	865	921	1786	51,57	48,43
HTTP+File transfer	0	15	2*h	RPROP	1760	26	1786	98,54	1,46
HTTP+Port scan	0	15	2*h	RPROP	1722	64	1786	96,42	3,58
HTTP+non periodic snap.	0	15	2*h	RPROP	1683	103	1786	94,23	5,77
HTTP+periodic snap.	0	15	2*h	RPROP	1726	60	1786	96,64	3,36
HTTP Licit	0	15	2*h	QuickPROP	775	1011	1786	56,61	43,39
HTTP+File transfer	0	15	2*h	QuickPROP	1757	29	1786	98,38	1,62
HTTP+Port scan	0	15	2*h	QuickPROP	1730	56	1786	96,86	3,14
HTTP+non periodic snap.	0	15	2*h	QuickPROP	1656	130	1786	92,72	7,28
HTTP+periodic snap.	0	15	2*h	QuickPROP	1706	80	1786	95,52	4,48
HTTP Licit	0	5	2*h	Batch	1796	0	1796	0,00	100,00
HTTP+File transfer	0	5	2*h	Batch	1796	0	1796	100,00	0,00
HTTP+Port scan	0	5	2*h	Batch	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	2*h	Batch	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	2*h	Batch	1796	0	1796	100,00	0,00
HTTP Licit	0	5	2*h	Incremental	693	1103	1796	61,41	38,59
HTTP+File transfer	0	5	2*h	Incremental	1749	47	1796	97,38	2,62
HTTP+Port scan	0	5	2*h	Incremental	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	2*h	Incremental	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	2*h	Incremental	1796	0	1796	100,00	0,00
HTTP Licit	0	5	2*h	RPROP	691	1105	1796	61,53	38,47

HTTP+File transfer	0	5	2*h	RPROP	1749	47	1796	97,38	2,62
HTTP+Port scan	0	5	2*h	RPROP	1796	0	1796	100,00	0,00
HTTP+non periodic snap.	0	5	2*h	RPROP	1796	0	1796	100,00	0,00
HTTP+periodic snap.	0	5	2*h	RPROP	1796	0	1796	100,00	0,00
HTTP Licit	0	5	2*h	QuickPROP	652	1144	1796	63,70	36,30
HTTP+File transfer	0	5	2*h	QuickPROP	1724	72	1796	95,99	4,01
HTTP+Port scan	0	5	2*h	QuickPROP	1792	4	1796	99,78	0,22
HTTP+non periodic snap.	0	5	2*h	QuickPROP	1791	5	1796	99,72	0,28
HTTP+periodic snap.	0	5	2*h	QuickPROP	1793	3	1796	99,83	0,17
HTTP Licit	10	10	2*h	Batch	171	0	171	0,00	100,00
HTTP+File transfer	10	10	2*h	Batch	171	0	171	100,00	0,00
HTTP+Port scan	10	10	2*h	Batch	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	2*h	Batch	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	2*h	Batch	171	0	171	100,00	0,00
HTTP Licit	10	10	2*h	Incremental	171	0	171	0,00	100,00
HTTP+File transfer	10	10	2*h	Incremental	171	0	171	100,00	0,00
HTTP+Port scan	10	10	2*h	Incremental	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	2*h	Incremental	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	2*h	Incremental	171	0	171	100,00	0,00
HTTP Licit	10	10	2*h	RPROP	171	0	171	0,00	100,00
HTTP+File transfer	10	10	2*h	RPROP	171	0	171	100,00	0,00
HTTP+Port scan	10	10	2*h	RPROP	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	2*h	RPROP	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	2*h	RPROP	171	0	171	100,00	0,00
HTTP Licit	10	10	2*h	QuickPROP	171	0	171	0,00	100,00
HTTP+File transfer	10	10	2*h	QuickPROP	171	0	171	100,00	0,00
HTTP+Port scan	10	10	2*h	QuickPROP	171	0	171	100,00	0,00
HTTP+non periodic snap.	10	10	2*h	QuickPROP	171	0	171	100,00	0,00
HTTP+periodic snap.	10	10	2*h	QuickPROP	171	0	171	100,00	0,00
HTTP Licit	10	15	2*h	Batch	166	0	166	0,00	100,00
HTTP+File transfer	10	15	2*h	Batch	166	0	166	100,00	0,00
HTTP+Port scan	10	15	2*h	Batch	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	2*h	Batch	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	2*h	Batch	166	0	166	100,00	0,00
HTTP Licit	10	15	2*h	Incremental	166	0	166	0,00	100,00
HTTP+File transfer	10	15	2*h	Incremental	166	0	166	100,00	0,00
HTTP+Port scan	10	15	2*h	Incremental	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	2*h	Incremental	166	0	166	100,00	0,00
HTTP+periodic snap.	10	15	2*h	Incremental	166	0	166	100,00	0,00
HTTP Licit	10	15	2*h	RPROP	166	0	166	0,00	100,00
HTTP+File transfer	10	15	2*h	RPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	2*h	RPROP	166	0	166	100,00	0,00
HTTP+non periodic snap.	10	15	2*h	RPROP	166	0	166	100,00	0,00

HTTP+periodic snap.	10	15	2*h	RPROP	166	0	166	100,00	0,00
HTTP Licit	10	15	2*h	QuickPROP	124	42	166	25,30	74,70
HTTP+File transfer	10	15	2*h	QuickPROP	166	0	166	100,00	0,00
HTTP+Port scan	10	15	2*h	QuickPROP	162	4	166	97,59	2,41
HTTP+non periodic snap.	10	15	2*h	QuickPROP	151	15	166	90,96	9,04
HTTP+periodic snap.	10	15	2*h	QuickPROP	163	3	166	98,19	1,81
HTTP Licit	10	5	2*h	Batch	176	0	176	0,00	100,00
HTTP+File transfer	10	5	2*h	Batch	176	0	176	100,00	0,00
HTTP+Port scan	10	5	2*h	Batch	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	2*h	Batch	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	2*h	Batch	176	0	176	100,00	0,00
HTTP Licit	10	5	2*h	Incremental	176	0	176	0,00	100,00
HTTP+File transfer	10	5	2*h	Incremental	176	0	176	100,00	0,00
HTTP+Port scan	10	5	2*h	Incremental	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	2*h	Incremental	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	2*h	Incremental	176	0	176	100,00	0,00
HTTP Licit	10	5	2*h	RPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	2*h	RPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	2*h	RPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	2*h	RPROP	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	2*h	RPROP	176	0	176	100,00	0,00
HTTP Licit	10	5	2*h	QuickPROP	176	0	176	0,00	100,00
HTTP+File transfer	10	5	2*h	QuickPROP	176	0	176	100,00	0,00
HTTP+Port scan	10	5	2*h	QuickPROP	176	0	176	100,00	0,00
HTTP+non periodic snap.	10	5	2*h	QuickPROP	176	0	176	100,00	0,00
HTTP+periodic snap.	10	5	2*h	QuickPROP	176	0	176	100,00	0,00
HTTP Licit	2	10	2*h	Batch	891	0	891	0,00	100,00
HTTP+File transfer	2	10	2*h	Batch	891	0	891	100,00	0,00
HTTP+Port scan	2	10	2*h	Batch	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	2*h	Batch	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	2*h	Batch	891	0	891	100,00	0,00
HTTP Licit	2	10	2*h	Incremental	778	113	891	12,68	87,32
HTTP+File transfer	2	10	2*h	Incremental	890	1	891	99,89	0,11
HTTP+Port scan	2	10	2*h	Incremental	891	0	891	100,00	0,00
HTTP+non periodic snap.	2	10	2*h	Incremental	891	0	891	100,00	0,00
HTTP+periodic snap.	2	10	2*h	Incremental	891	0	891	100,00	0,00
HTTP Licit	2	10	2*h	RPROP	434	457	891	51,29	48,71
HTTP+File transfer	2	10	2*h	RPROP	882	9	891	98,99	1,01
HTTP+Port scan	2	10	2*h	RPROP	863	28	891	96,86	3,14
HTTP+non periodic snap.	2	10	2*h	RPROP	839	52	891	94,16	5,84
HTTP+periodic snap.	2	10	2*h	RPROP	852	39	891	95,62	4,38
HTTP Licit	2	10	2*h	QuickPROP	686	205	891	23,01	76,99
HTTP+File transfer	2	10	2*h	QuickPROP	888	3	891	99,66	0,34

HTTP+Port scan	2	10	2*h	QuickPROP	883	8	891	99,10	0,90
HTTP+non periodic snap.	2	10	2*h	QuickPROP	873	18	891	97,98	2,02
HTTP+periodic snap.	2	10	2*h	QuickPROP	880	11	891	98,77	1,23
HTTP Licit	2	15	2*h	Batch	886	0	886	0,00	100,00
HTTP+File transfer	2	15	2*h	Batch	886	0	886	100,00	0,00
HTTP+Port scan	2	15	2*h	Batch	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	2*h	Batch	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	2*h	Batch	886	0	886	100,00	0,00
HTTP Licit	2	15	2*h	Incremental	886	0	886	0,00	100,00
HTTP+File transfer	2	15	2*h	Incremental	886	0	886	100,00	0,00
HTTP+Port scan	2	15	2*h	Incremental	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	2*h	Incremental	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	2*h	Incremental	886	0	886	100,00	0,00
HTTP Licit	2	15	2*h	RPROP	613	273	886	30,81	69,19
HTTP+File transfer	2	15	2*h	RPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	2*h	RPROP	871	15	886	98,31	1,69
HTTP+non periodic snap.	2	15	2*h	RPROP	855	31	886	96,50	3,50
HTTP+periodic snap.	2	15	2*h	RPROP	861	25	886	97,18	2,82
HTTP Licit	2	15	2*h	QuickPROP	886	0	886	0,00	100,00
HTTP+File transfer	2	15	2*h	QuickPROP	886	0	886	100,00	0,00
HTTP+Port scan	2	15	2*h	QuickPROP	886	0	886	100,00	0,00
HTTP+non periodic snap.	2	15	2*h	QuickPROP	886	0	886	100,00	0,00
HTTP+periodic snap.	2	15	2*h	QuickPROP	886	0	886	100,00	0,00
HTTP Licit	2	5	2*h	Batch	896	0	896	0,00	100,00
HTTP+File transfer	2	5	2*h	Batch	896	0	896	100,00	0,00
HTTP+Port scan	2	5	2*h	Batch	896	0	896	100,00	0,00
HTTP+non periodic snap.	2	5	2*h	Batch	896	0	896	100,00	0,00
HTTP+periodic snap.	2	5	2*h	Batch	896	0	896	100,00	0,00
HTTP Licit	2	5	2*h	Incremental	446	450	896	50,22	49,78
HTTP+File transfer	2	5	2*h	Incremental	879	17	896	98,10	1,90
HTTP+Port scan	2	5	2*h	Incremental	885	11	896	98,77	1,23
HTTP+non periodic snap.	2	5	2*h	Incremental	875	21	896	97,66	2,34
HTTP+periodic snap.	2	5	2*h	Incremental	877	19	896	97,88	2,12
HTTP Licit	2	5	2*h	RPROP	280	616	896	68,75	31,25
HTTP+File transfer	2	5	2*h	RPROP	868	28	896	96,88	3,13
HTTP+Port scan	2	5	2*h	RPROP	859	37	896	95,87	4,13
HTTP+non periodic snap.	2	5	2*h	RPROP	842	54	896	93,97	6,03
HTTP+periodic snap.	2	5	2*h	RPROP	853	43	896	95,20	4,80
HTTP Licit	2	5	2*h	QuickPROP	406	490	896	54,69	45,31
HTTP+File transfer	2	5	2*h	QuickPROP	876	20	896	97,77	2,23
HTTP+Port scan	2	5	2*h	QuickPROP	880	16	896	98,21	1,79
HTTP+non periodic snap.	2	5	2*h	QuickPROP	872	24	896	97,32	2,68
HTTP+periodic snap.	2	5	2*h	QuickPROP	877	19	896	97,88	2,12

HTTP Licit	5	10	2*h	Batch	351	0	351	0,00	100,00
HTTP+File transfer	5	10	2*h	Batch	351	0	351	100,00	0,00
HTTP+Port scan	5	10	2*h	Batch	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	2*h	Batch	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	2*h	Batch	351	0	351	100,00	0,00
HTTP Licit	5	10	2*h	Incremental	351	0	351	0,00	100,00
HTTP+File transfer	5	10	2*h	Incremental	351	0	351	100,00	0,00
HTTP+Port scan	5	10	2*h	Incremental	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	2*h	Incremental	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	2*h	Incremental	351	0	351	100,00	0,00
HTTP Licit	5	10	2*h	RPROP	349	2	351	0,57	99,43
HTTP+File transfer	5	10	2*h	RPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	2*h	RPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	2*h	RPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	2*h	RPROP	351	0	351	100,00	0,00
HTTP Licit	5	10	2*h	QuickPROP	351	0	351	0,00	100,00
HTTP+File transfer	5	10	2*h	QuickPROP	351	0	351	100,00	0,00
HTTP+Port scan	5	10	2*h	QuickPROP	351	0	351	100,00	0,00
HTTP+non periodic snap.	5	10	2*h	QuickPROP	351	0	351	100,00	0,00
HTTP+periodic snap.	5	10	2*h	QuickPROP	351	0	351	100,00	0,00
HTTP Licit	5	15	2*h	Batch	346	0	346	0,00	100,00
HTTP+File transfer	5	15	2*h	Batch	346	0	346	100,00	0,00
HTTP+Port scan	5	15	2*h	Batch	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	2*h	Batch	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	2*h	Batch	346	0	346	100,00	0,00
HTTP Licit	5	15	2*h	Incremental	346	0	346	0,00	100,00
HTTP+File transfer	5	15	2*h	Incremental	346	0	346	100,00	0,00
HTTP+Port scan	5	15	2*h	Incremental	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	2*h	Incremental	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	2*h	Incremental	346	0	346	100,00	0,00
HTTP Licit	5	15	2*h	RPROP	308	38	346	10,98	89,02
HTTP+File transfer	5	15	2*h	RPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	2*h	RPROP	333	13	346	96,24	3,76
HTTP+non periodic snap.	5	15	2*h	RPROP	336	10	346	97,11	2,89
HTTP+periodic snap.	5	15	2*h	RPROP	332	14	346	95,95	4,05
HTTP Licit	5	15	2*h	QuickPROP	346	0	346	0,00	100,00
HTTP+File transfer	5	15	2*h	QuickPROP	346	0	346	100,00	0,00
HTTP+Port scan	5	15	2*h	QuickPROP	346	0	346	100,00	0,00
HTTP+non periodic snap.	5	15	2*h	QuickPROP	346	0	346	100,00	0,00
HTTP+periodic snap.	5	15	2*h	QuickPROP	346	0	346	100,00	0,00
HTTP Licit	5	5	2*h	Batch	356	0	356	0,00	100,00
HTTP+File transfer	5	5	2*h	Batch	356	0	356	100,00	0,00
HTTP+Port scan	5	5	2*h	Batch	356	0	356	100,00	0,00

HTTP+non periodic snap.	5	5	2*h	Batch	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	2*h	Batch	356	0	356	100,00	0,00
HTTP Licit	5	5	2*h	Incremental	339	17	356	4,78	95,22
HTTP+File transfer	5	5	2*h	Incremental	356	0	356	100,00	0,00
HTTP+Port scan	5	5	2*h	Incremental	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	2*h	Incremental	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	2*h	Incremental	356	0	356	100,00	0,00
HTTP Licit	5	5	2*h	RPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	2*h	RPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	2*h	RPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	2*h	RPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	2*h	RPROP	356	0	356	100,00	0,00
HTTP Licit	5	5	2*h	QuickPROP	339	17	356	4,78	95,22
HTTP+File transfer	5	5	2*h	QuickPROP	356	0	356	100,00	0,00
HTTP+Port scan	5	5	2*h	QuickPROP	356	0	356	100,00	0,00
HTTP+non periodic snap.	5	5	2*h	QuickPROP	356	0	356	100,00	0,00
HTTP+periodic snap.	5	5	2*h	QuickPROP	356	0	356	100,00	0,00